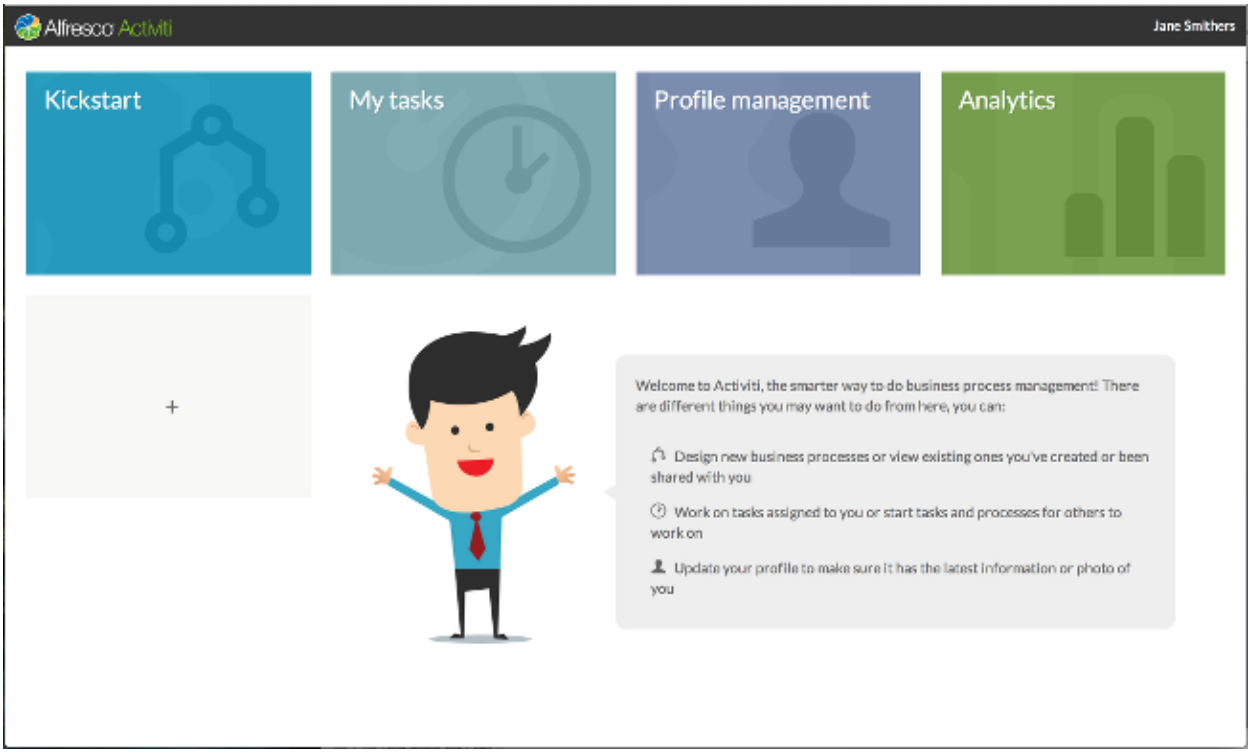
Alfresco Activiti is a BPM platform that can be run on-premise or hosted on a private or public cloud, single or multi-tenant.

Once you have registered and signed in for the first time, you'll see your **Landing Page** with some helpful hints from James.

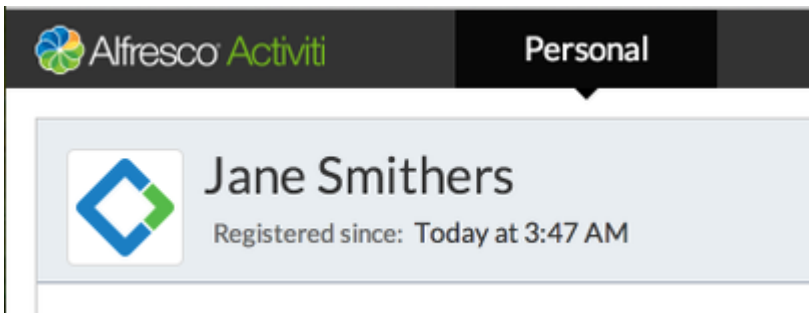
1. Introduction



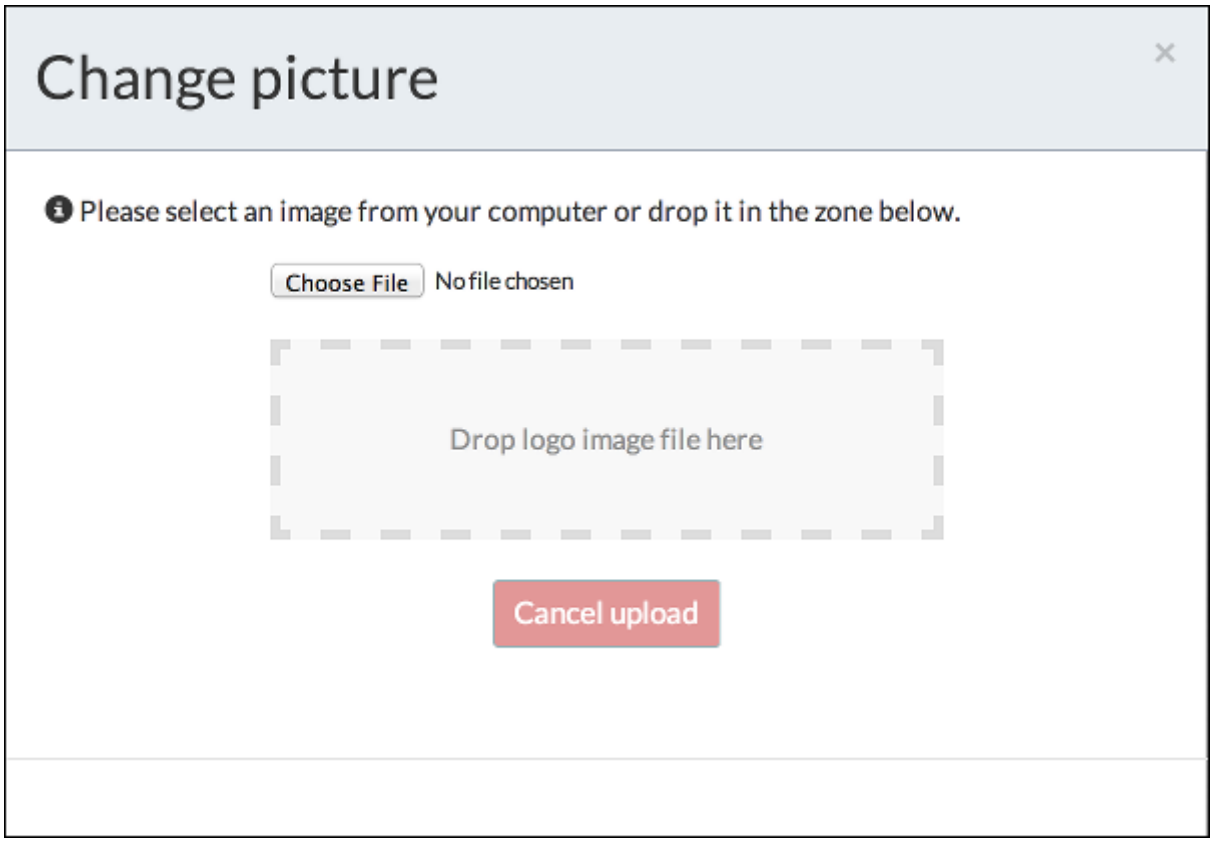
1.1. Personal profile

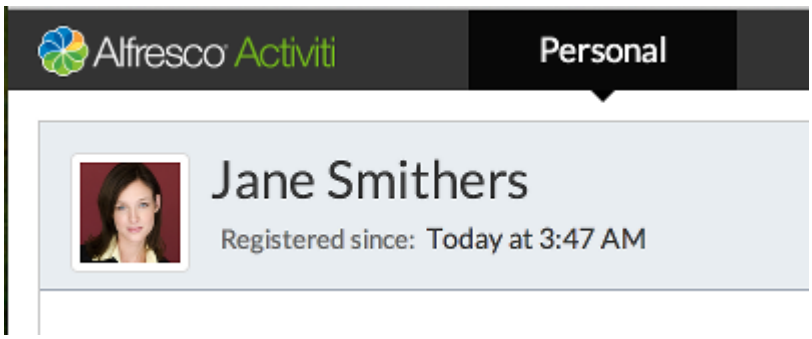
The first thing you might want to do is add a photo to your profile, so click **Profile management**.

On the **Personal** page you can edit your details, such as your name, change your password, and view your group membership and capabilities.



To add your photo, click the image to the left of your name and upload the desired photo.





1.2. Navigation

You can click on the Alfresco Activiti logo at any time to return to your landing page. Now that you've got your photo uploaded, go back to the landing page where you can do the following:

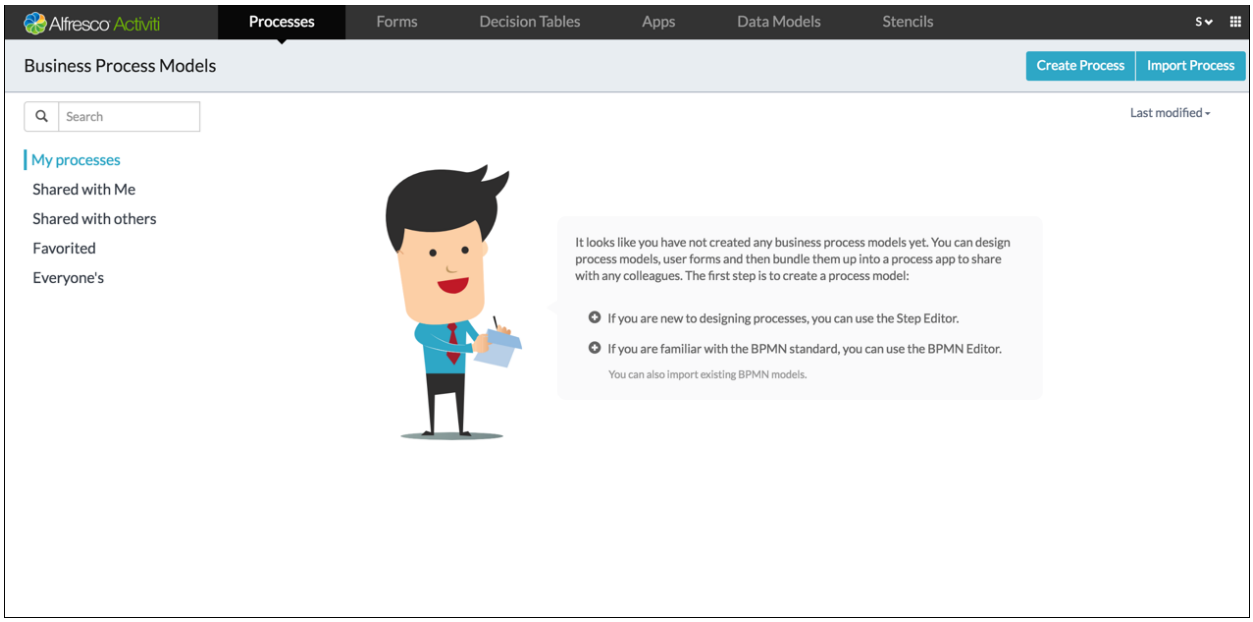
- **Kickstart** - Design your process
- **My Tasks** - View your task inbox or queue
- **Profile management** / **Identity management** - Manage user and group capabilities
- **Analytics** - Generate reports on process performance

Depending on the capabilities of your account you may or may not get access to **Kickstart** or **Analytics**. For example, if you have administrator capabilities, then **Profile management** will be displayed as **Identity management**. Use this tile to access your profile page as well as to manage user, group, and capability management pages for your tenant or the whole system.

1.3. Using Kickstart App

Use the **Kickstart App** tile to get started with your process design. The following tabs are available on the header:

- Processes
- Forms
- Decision Tables
- Apps
- Data Models
- Stencils



By default, you'll be directed to the **Process** page. You can create a new process or edit an existing process model. If your account has the capability, you can also import existing models that are defined in BPMN 2.0 standard format.

1.4. Creating a process model

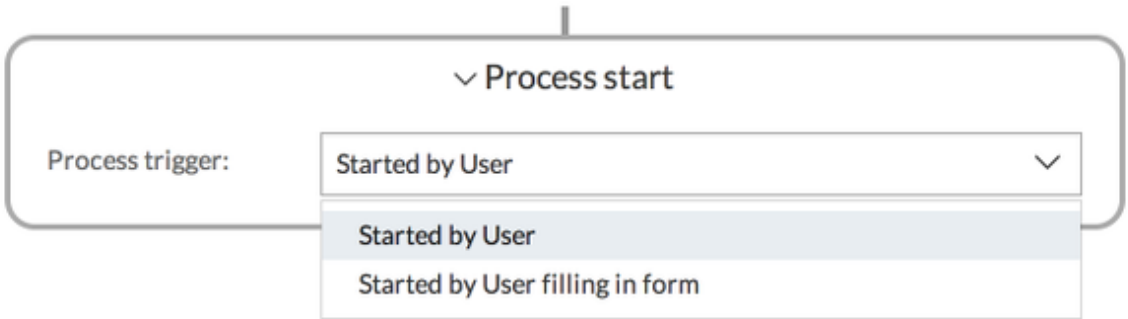
You can create a process model using the **Step Editor** or **BPMN Editor**. While **Step Editor** enables you to define a business process through a sequence of steps, **BPMN Editor** is a more powerful process design tool for creating BPMN 2.0 standard models.

Let's start by creating a process model using the **Step Editor**:

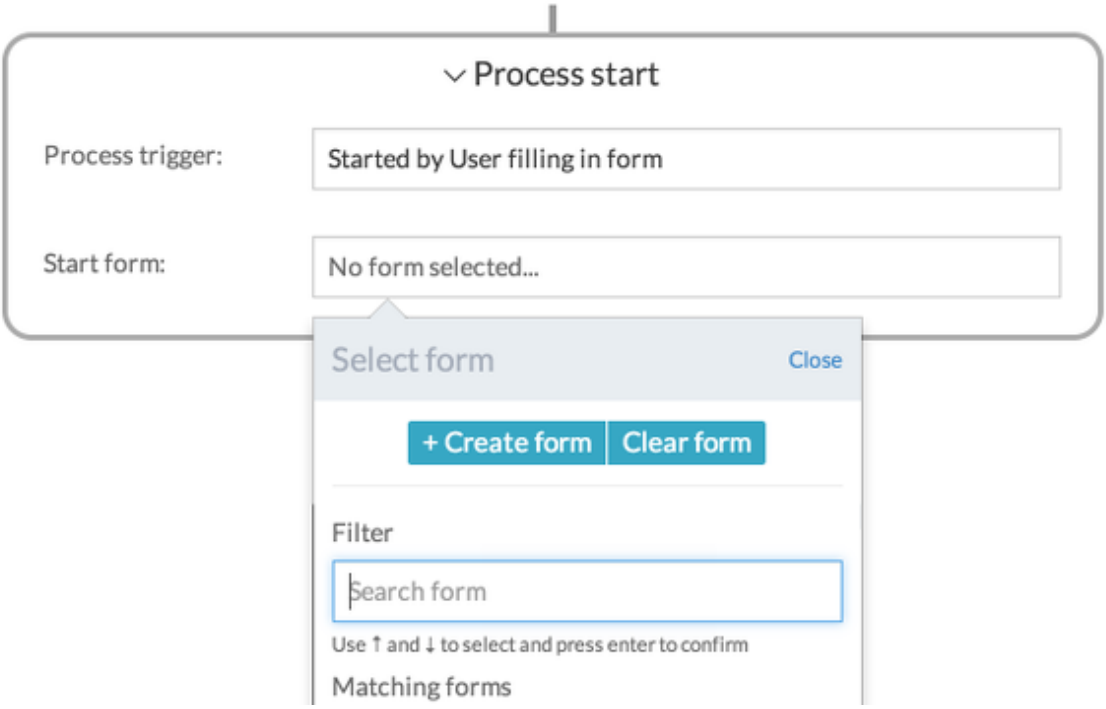
1. Click the hint box next to James, or **Create Process**. The **Create a new business process model** dialog box appears.
2. Type the **Name** and **Description** of your process Model.
3. Select the **Editor type** and **Stencil**.
4. Click **Create new model**. The **Step Editor** page is displayed.

   By default, **Step Editor** includes a number of **Steps**, however this depends on the **Stencil** that you selected for editing the process model.

5. Click **Process start** to expand and start by setting the process trigger to User filling a form.



6. Click **Create form** to create a new form or select an existing form from your **Forms** library. The **Form Editor** is displayed.



Note: Any form that's created this way will not be available in your Forms library because it was created as part of this process model. To create a form that you can reuse in other process models, you must create it from the main **Forms** page. In this example, the form is defined in the **Step Editor**.

The Forms editor has the following tabs:

- **Design** - Define the layout of form fields from the palette.
- **Tabs** - Customize tab names to display in the form.
- **Outcomes** - Define the outcome buttons for the form.
- **Style** - Define the style (css) for the form elements. For example, adding the following style in the Style panel will convert the field background to blue:

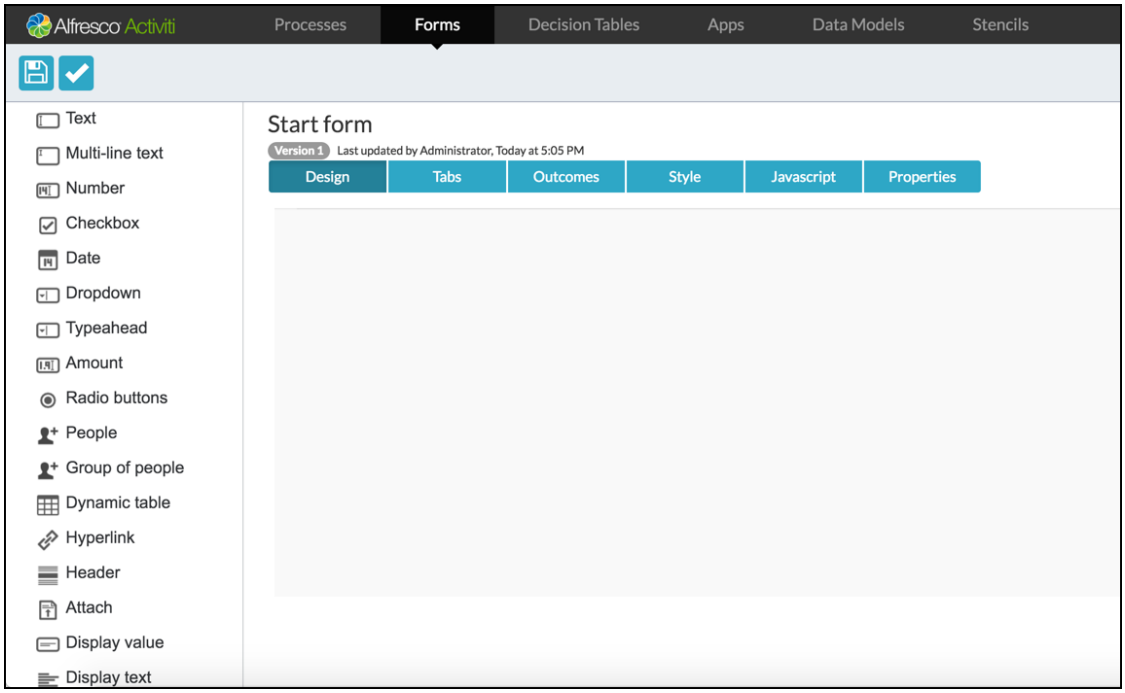
```
    .fields {
    background-color: blue;
}
```

- **Javascript** - Define javascript code for an element in the form. For example:

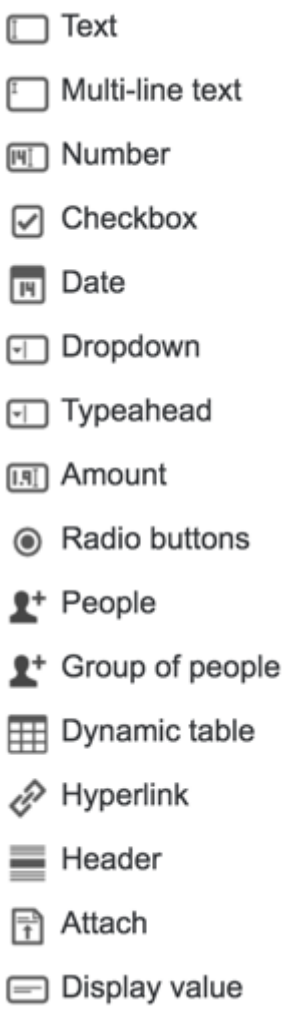
```
// __var currentUser = scope.$root.account;__
   __console.log(currentUser);__
  __alert ("Hello World!");__
```

- **Properties** - Define custom properties (metadata) for the form. This is particularly useful when using a custom form renderer (Jave API or Rest API) to retrieve the properties.

- **Variables** - Define variables in the form.



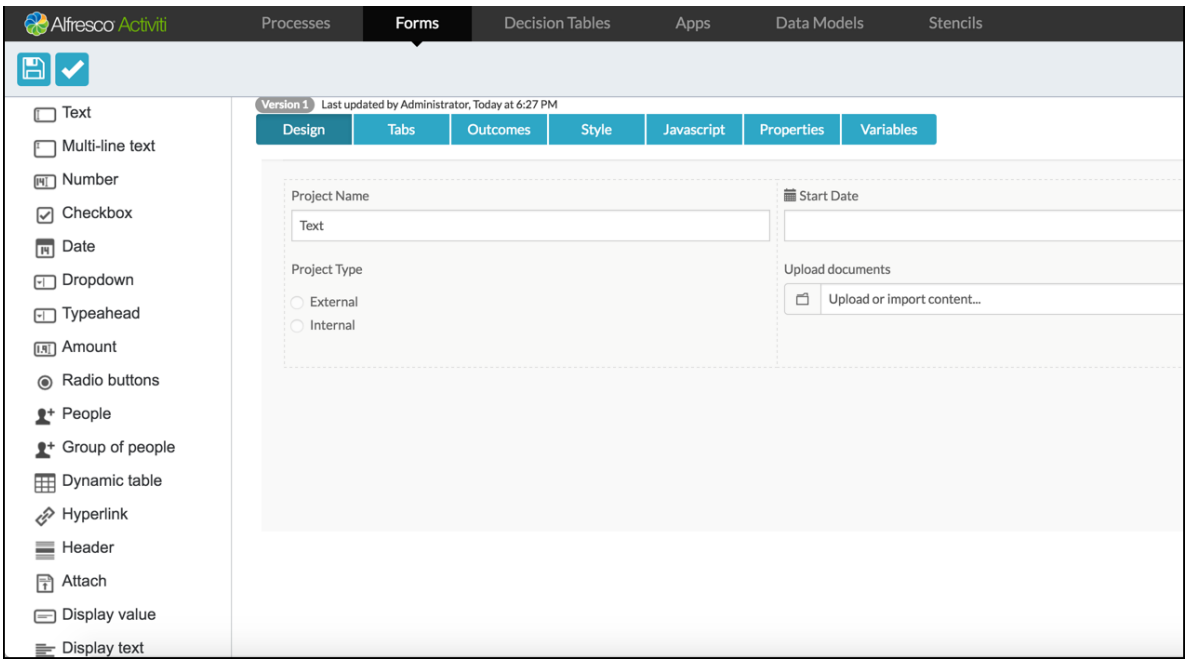
You can design the form layout by simply dragging and dropping the required field type from the palette on the left to the form editor.



For each field dropped in the **Design** area, you can hover over it and edit the field properties using the pencil icon. Alternatively, click **X** to remove a field from the form.

Note: The options that become available in the edit view are determined by the field type selected from the palette. For example, a checkbox field has General, Visibility, and Style tabs, whereas a radio button field type might have an addtional tab called Options.

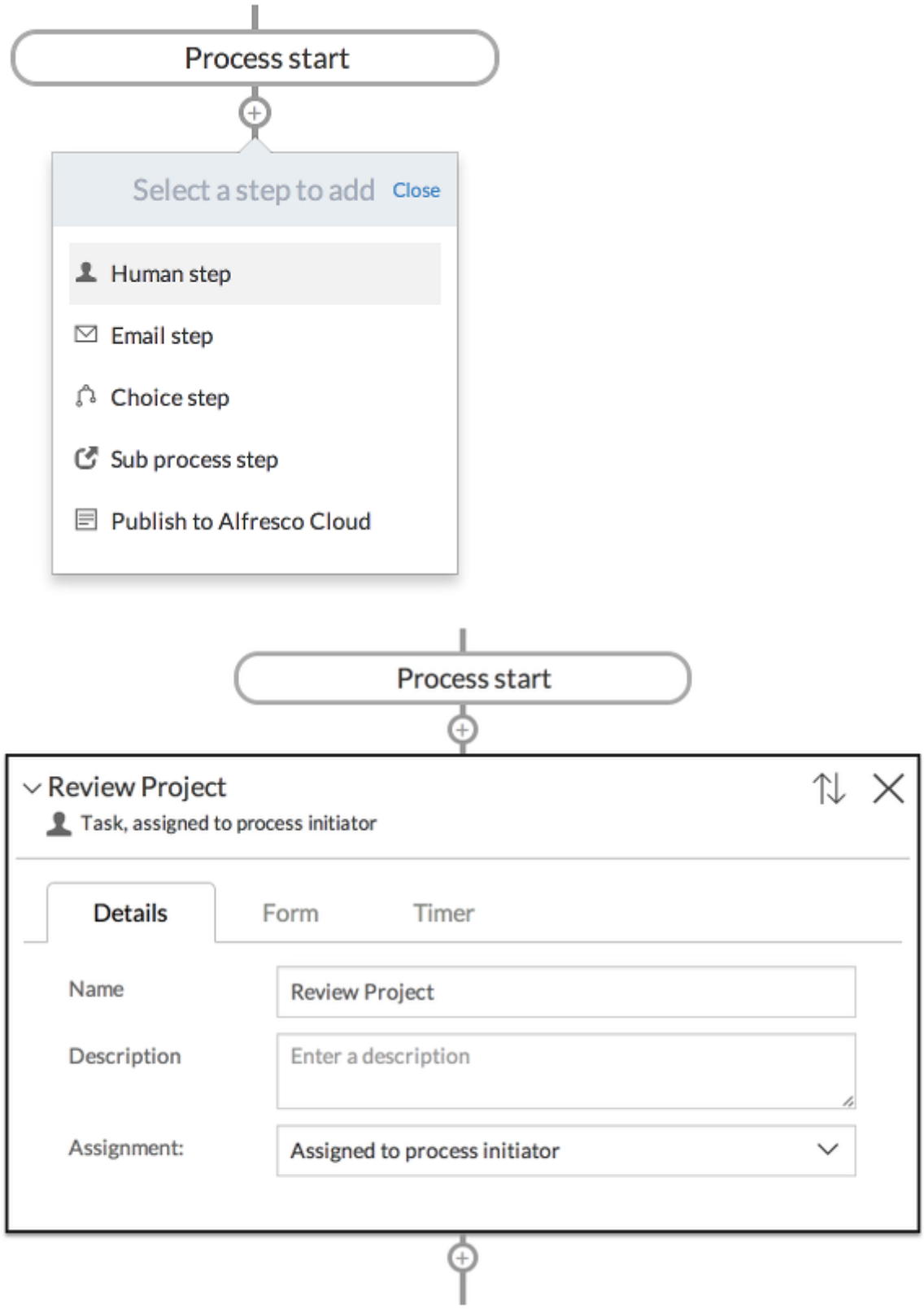
Add labels for the selected fields. Optionally, you can reference a display label with the value entered by a user running the process. You can also define if the field is required to be filled before the form can be completed.



7. When you've finished designing the form, click **Save**. You'll be returned to the **Step Editor**.

8. Click the **+** (plus) icon at the bottom of the **Process start** box to add the first step in your process. The steps available to you are defined by the **Stencil** you associated the model with. The default stencil includes a **Human step** that can be used to assign a task to the user.

9. Select the **Human step** and fill in a name within the step box that you just created.



You can also specify who this task should be assigned to. For example:

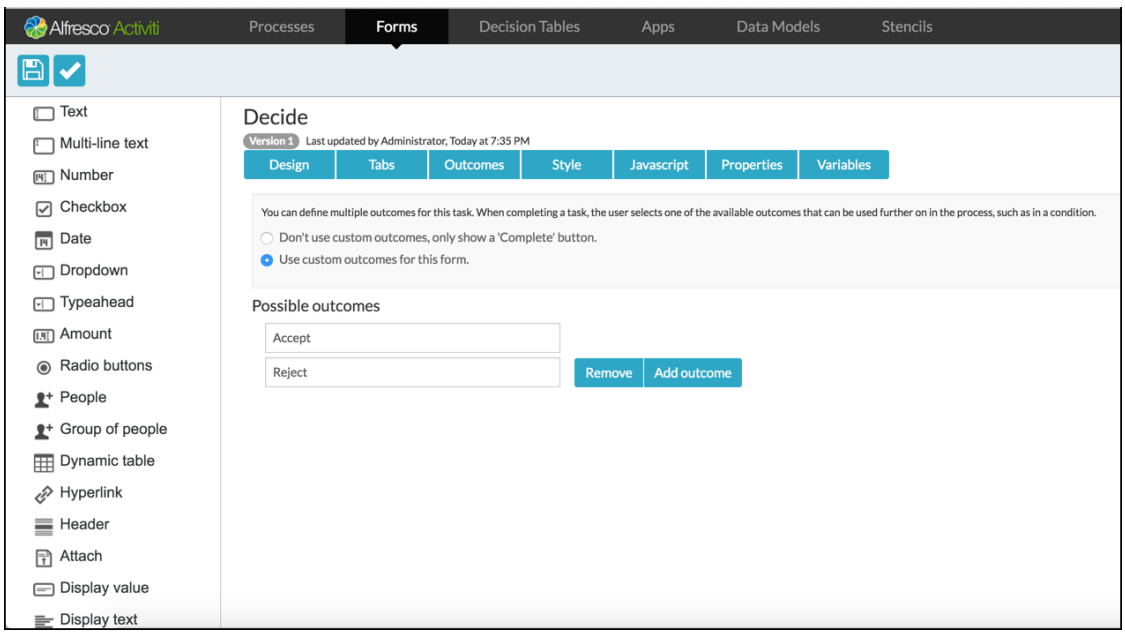
- Someone who initiated the process
- A single user
- A set of candidate users or depending on the type of account, a group of users.

Note: When a task is assigned to a group or a list of candidate users, all of those users can see the task in their tasks list, however to complete the task they must claim it first.

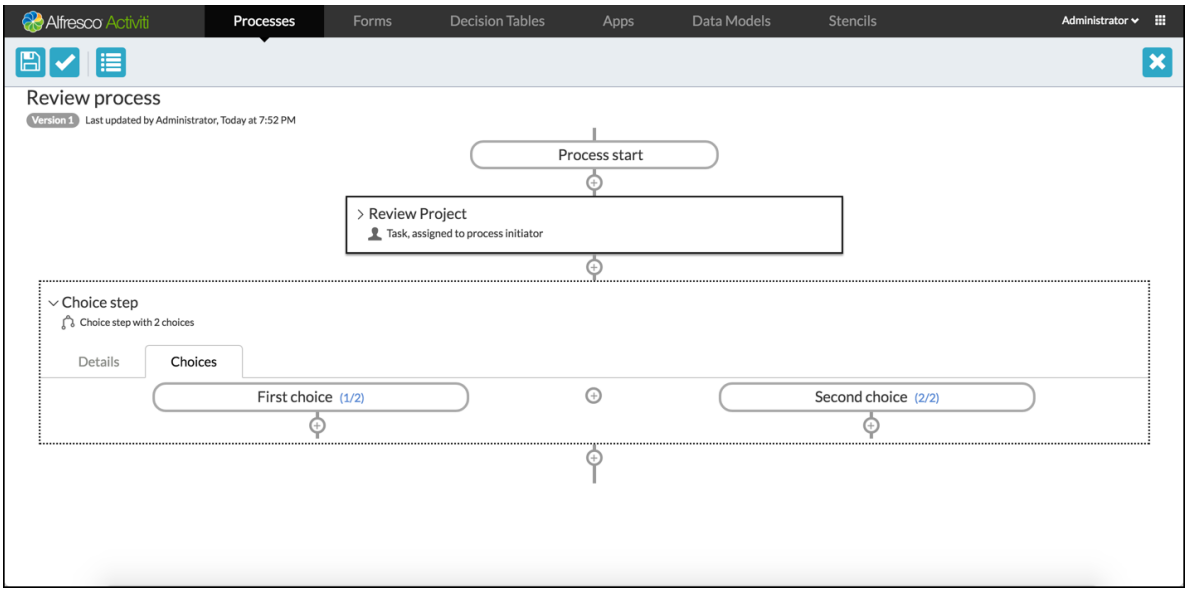
1.4.1. Assigning tasks

To simplify the process, we'll assign all tasks to the process initiator so that you can run the process and have the tasks assigned to yourself.

1. Click **Forms** > **Create Form**. The Create a new form dialog box appears.
2. Enter a form name and click **Save**.
3. Drag a multiline text field and drop it to the form. Name the label as **Review comment**.
4. Click the **Outcomes** tab and then select **Use custom outcomes for this form**.
5. In **possible outcomes**, add the following outcomes and then save the form:
    - **Accept**
    - **Reject**

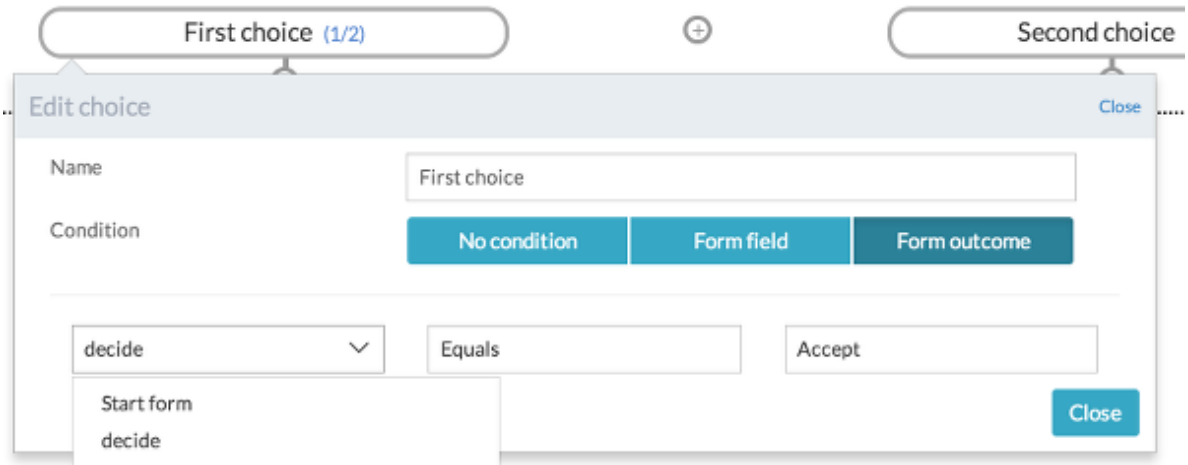
    The next step depends on the outcome selected in the previous step.
6. Add a **Choice step** by clicking the **+** (plus) icon below the **Review Project** step.



You can also add addtional choices by clicking the **+** (plus) icon in the center of the **Choice step**.

7. Click the relevant choice box to set the condition for the selected choice. The Edit choice dialog appears where you can select the condition based on the existing form fields or outcomes.

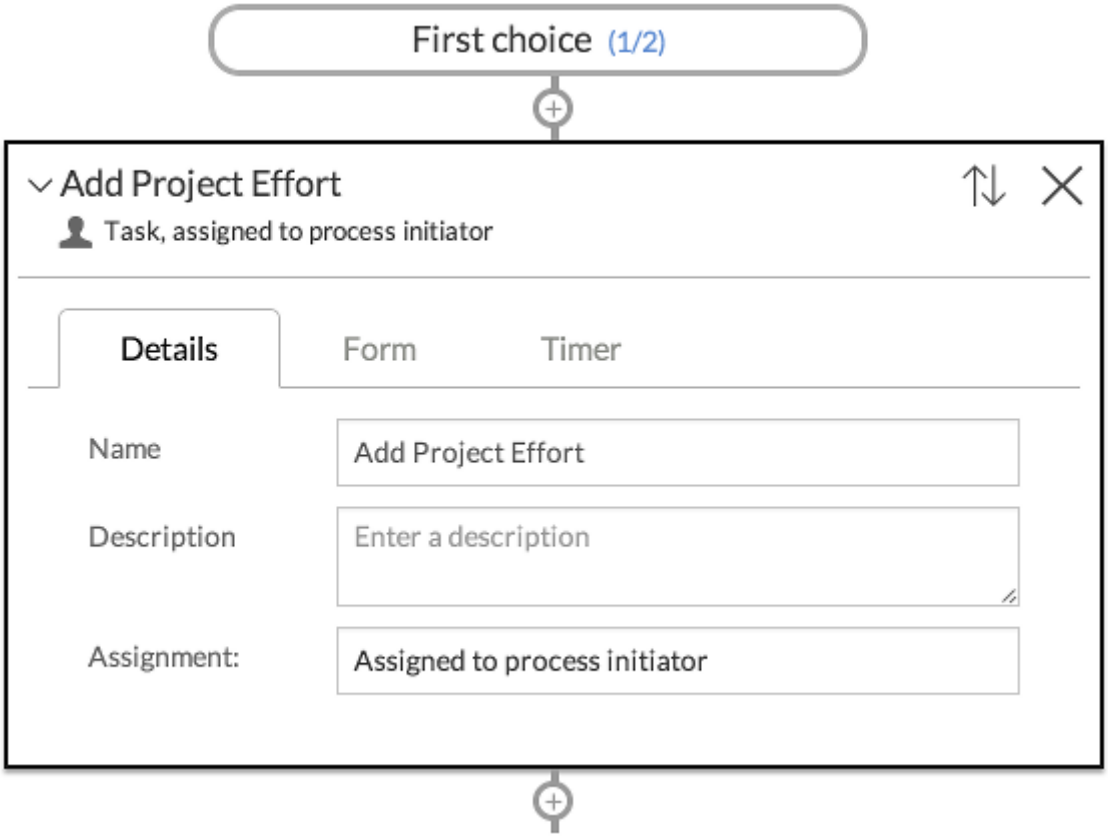
8. For the first choice, click **Form Outcome** and select the following values: **Review** form > **Equal** > **Accept**.



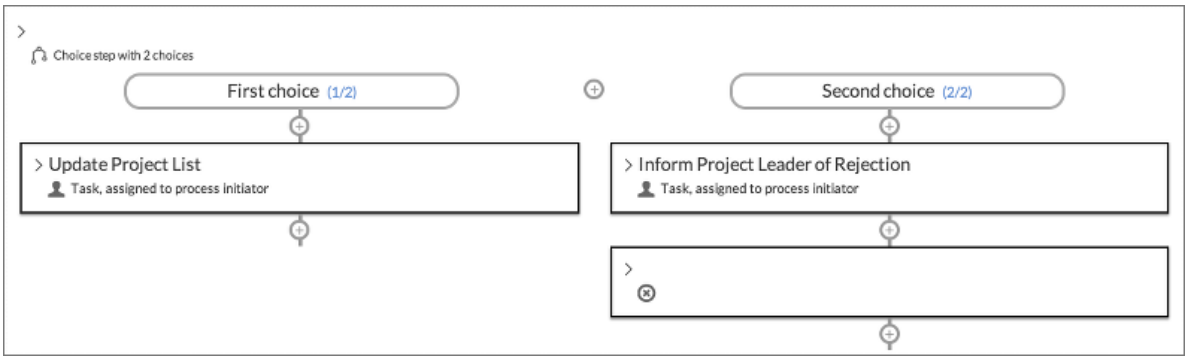
9. Click **Save**. Repeat the same for second choice: **Review form** > **Equal** > **Reject**.

   Note: Provide a meaningful name for the choice steps if you can.

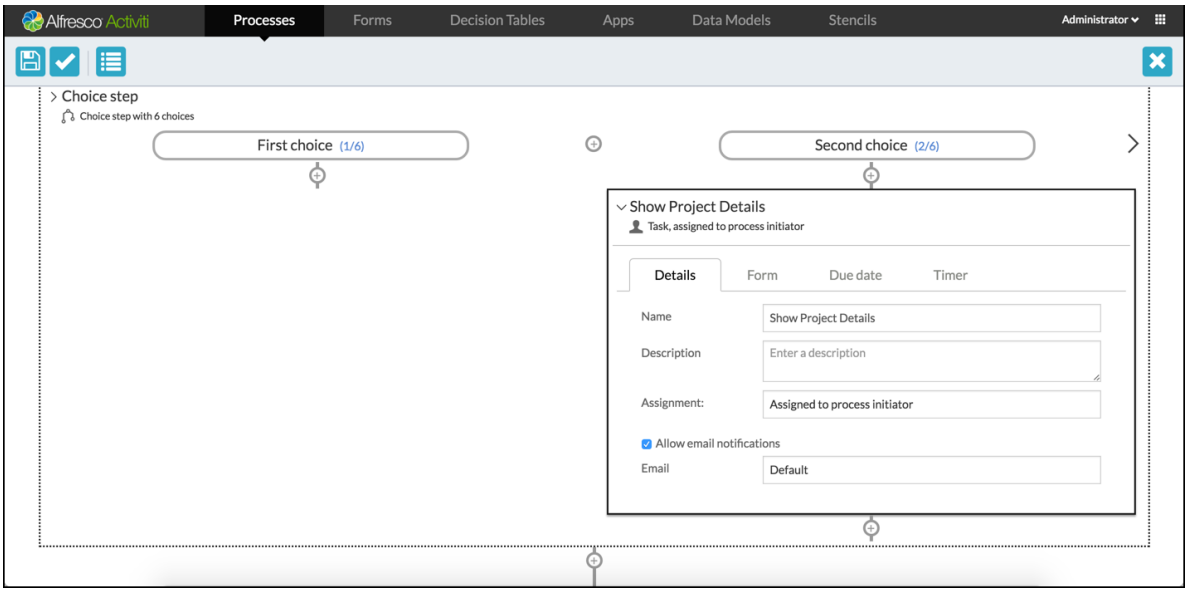
10. Add a task that should be done once the project review is accepted by clicking the + under the **First choice** box.



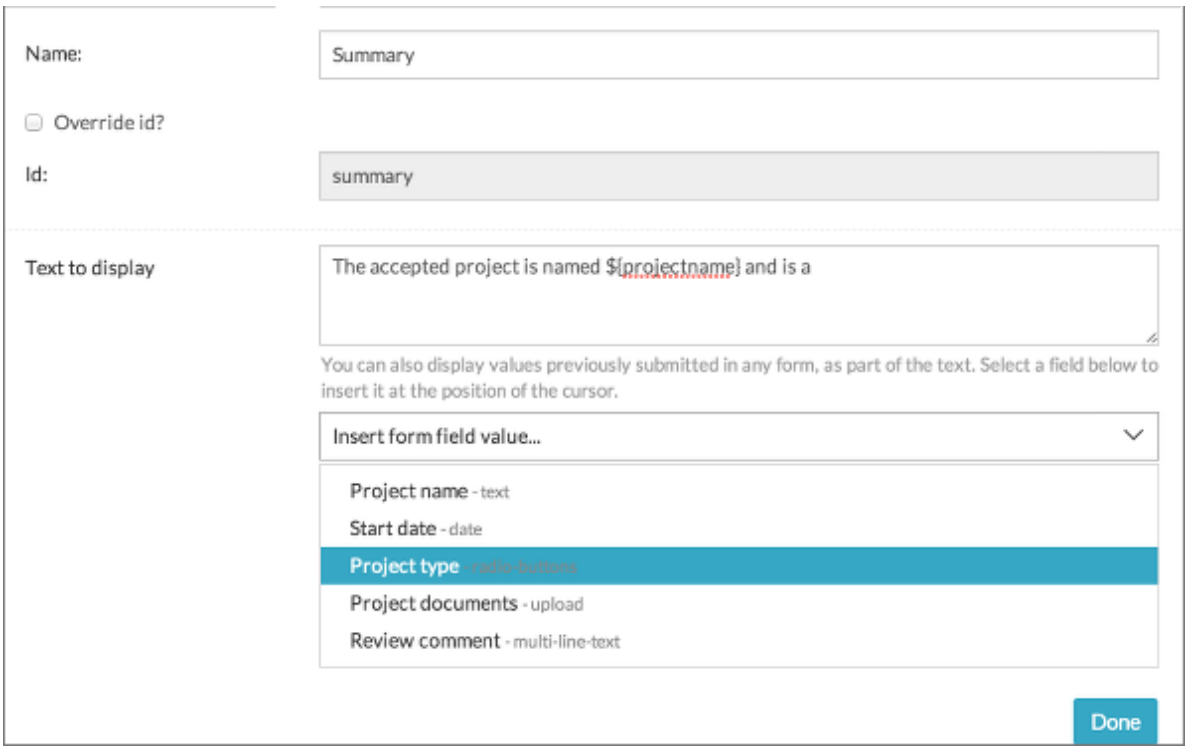
11. Now, add a simple human task called **Update Project List**. Under the **Second choice** box, add a human task with a name **Inform Project Leader of Rejection**. The aim is for the process to stop when the rejection task is completed. Therefore, add a **Stop step** to the bottom of this task.



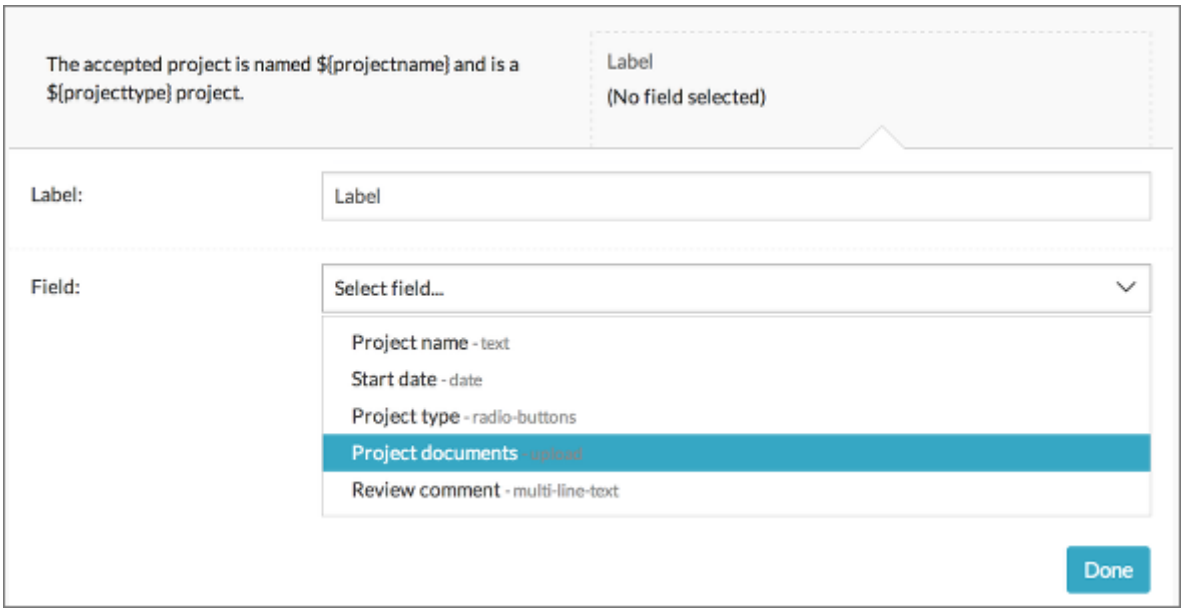
12. Continue with adding steps to the **First choice**, or in this case continue to add them after completing the Choice step by clicking the **+** at the very bottom. We'll just add a Human task with the name Show Project Details.



13. On the **Forms** tab for this task, create a new form. Drag a **Display text** field from the palette and enter the text message to display. The text can contain references to values added by a user in previous forms. There is a helper dropdown that you can select from to insert the given reference at the cursor position in the text.

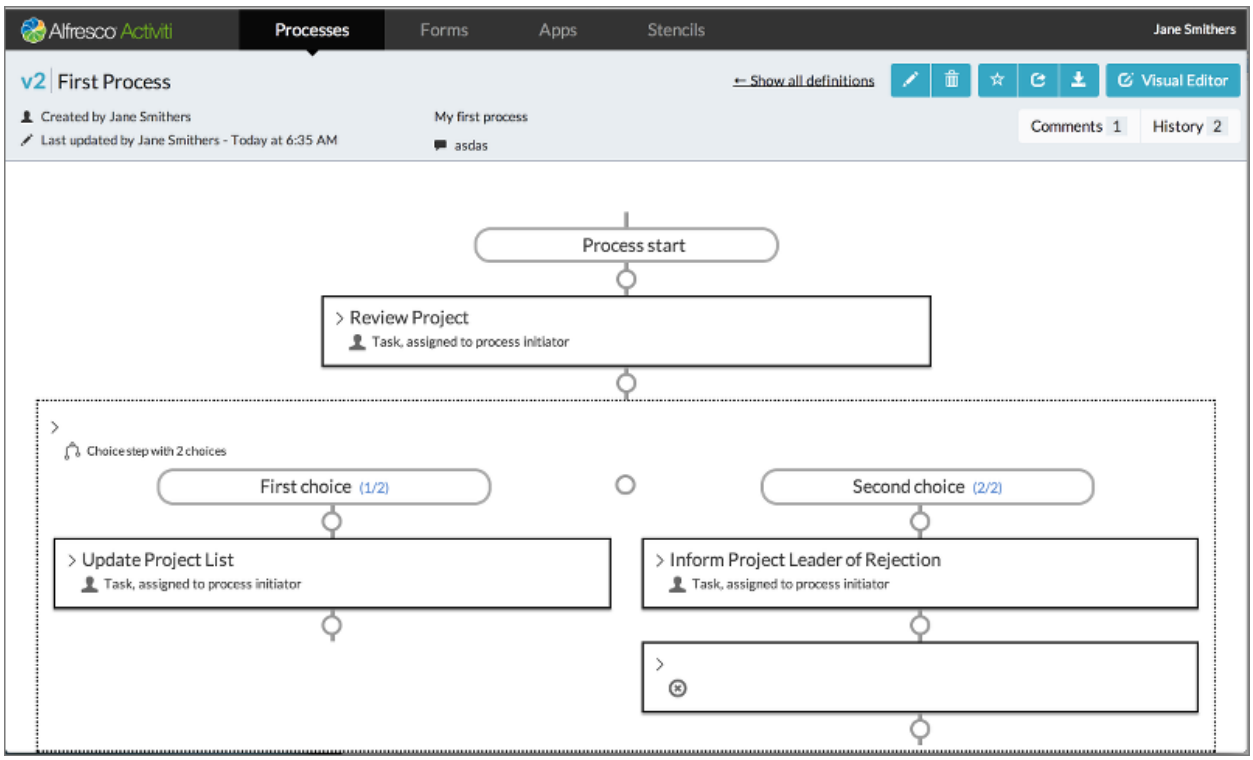


14. Add some text as shown. Then drag a **Display value** field from the palette and set it to display the project files by selecting the appropriate field from the list.



15. Save the form to return to the **Step Editor** and then save the process model you've just designed.

All your processes are listed with a thumbnail of the process. You can edit a process from the list by clicking **Visual Editor**. For any additional information about a model, click the thumbnail itself or the **Show Details** button on the top right corner of the thumbnail. This takes you to the **Details** page for the process model where you can see the preview model as well as the actions that you can perform on it.



Tips:

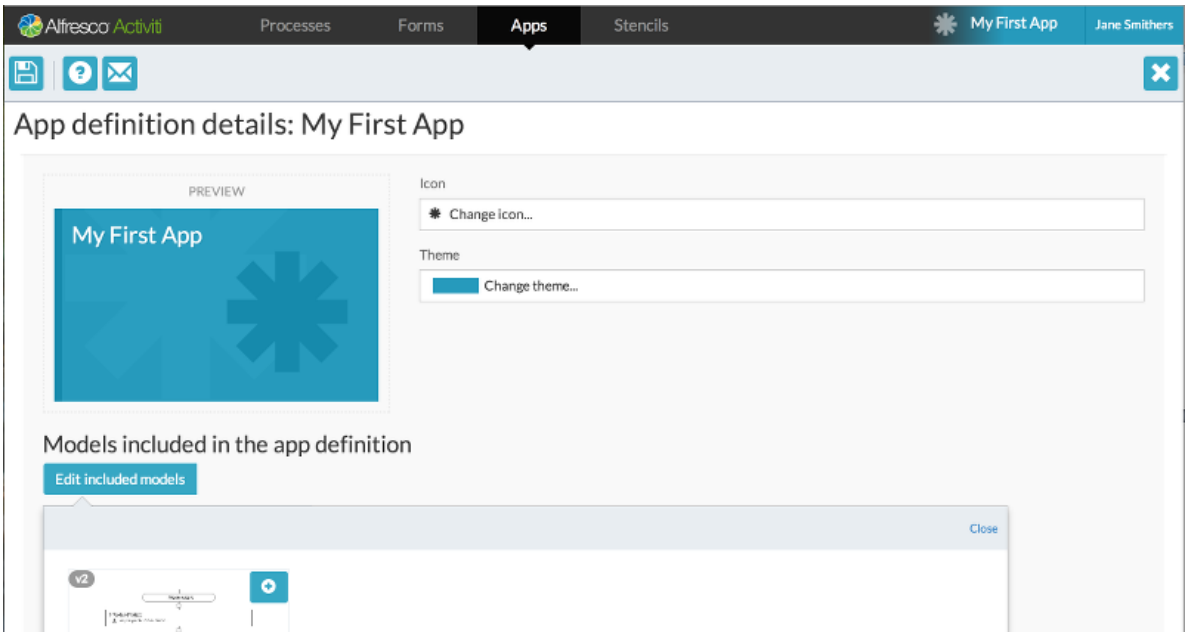
- When you edit and save a model, you can choose for the changes to be saved as a new version.

- Previous versions can be accessed from the **History** popup, as can any commentary from the **Comments** popup, where you can add further comments.
- Other action buttons are self-explanatory such as deleting, starring (favorites), sharing, and downloading the model.

1.4.2. Creating a process app

Now that we have a process defined, let's create a process app using the **Apps** page.

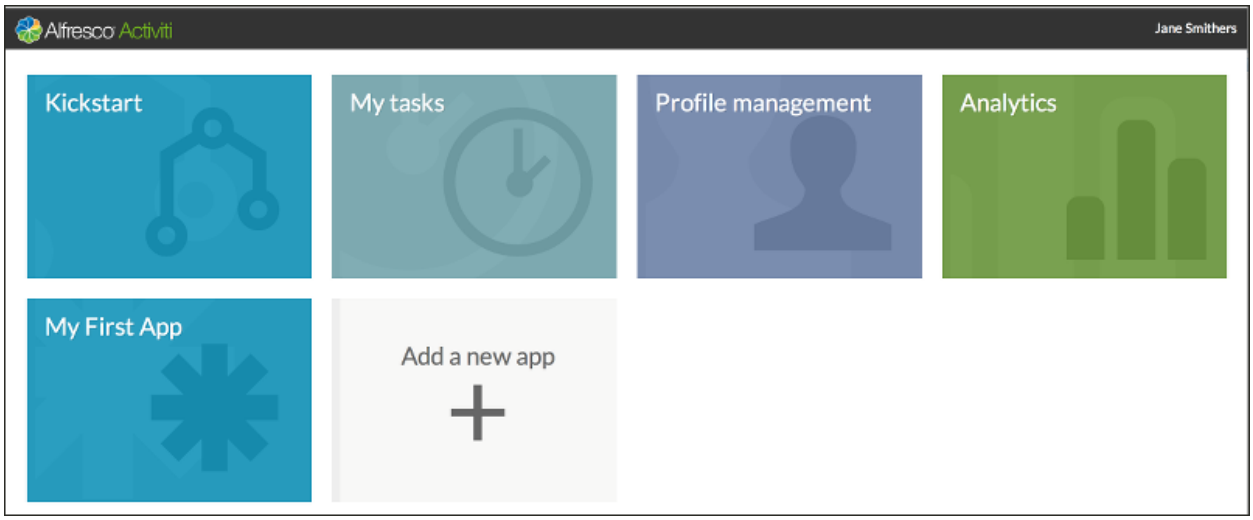
1. On the **Apps** page, click **Create app definition**.
2. Select an icon and theme for the app's tile. You can have an app without any process definitions linked to it, which lets you create a simple custom task list.
3. Click **Edit included models** to use the process we've just defined, and select from the lists to add a model.



4. Save the app and select the option to publish the app in the **Save** dialog to return the Apps list view.

   You can do similar actions on an app in its **Details** page for all models, such as deleting and sharing. You can also publish the app directly instead of doing it via the Save dialog. Publishing an app makes it available to everyone you've shared it with to add to their landing page. Let's add it to our landing page so we can see our process in action.

5. On your landing page, click the tile with the + (plus) icon. The Add app to landing page dialog appears.
6. Choose the apps you want to add and click **Deploy**. A new tile will be added to your landing page.



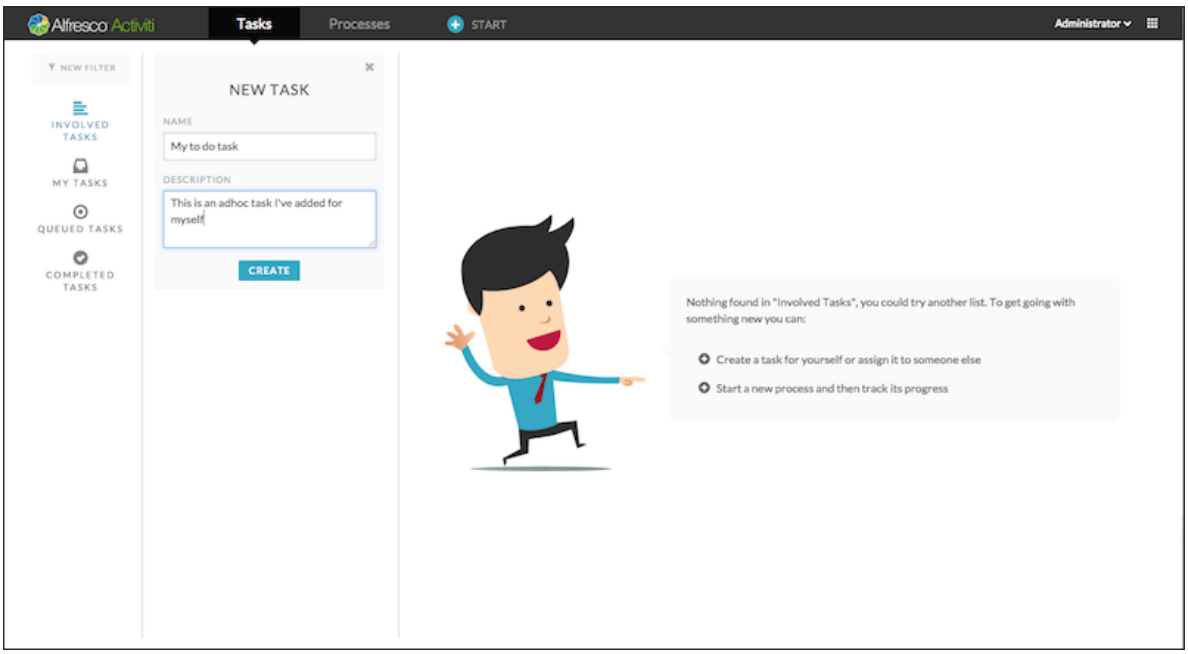
1.4.3. Using My Tasks and Process Apps

A process app is simply a collection of processes that you want to group together to make them available to yourself or other users you share it with.

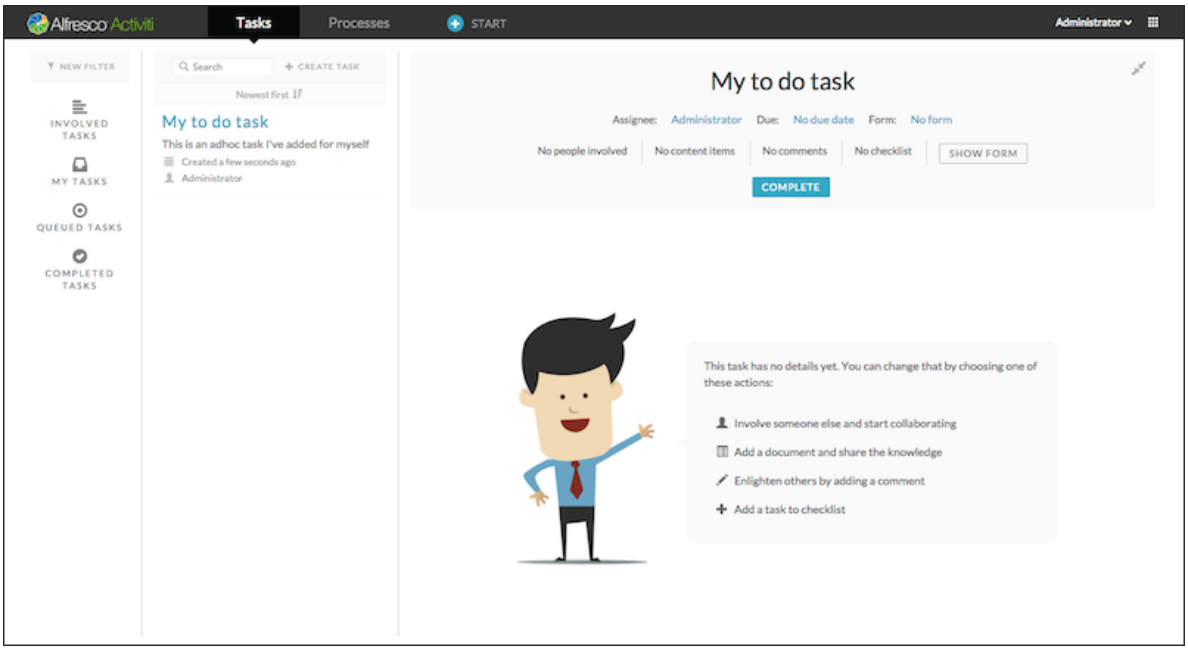
To access tasks and processes:

1. Click on the new app tile that you just created and you will be taken to its **Task** page. This will only show the tasks created within this app or as part of the processes from the app.

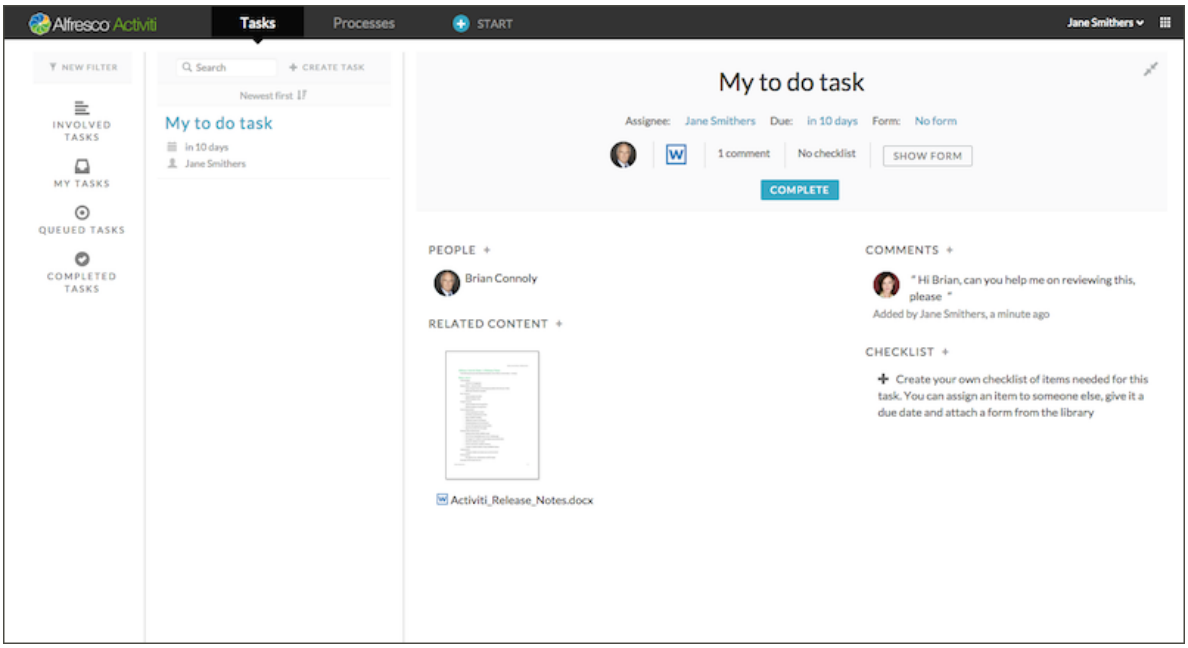
2. Click on the hint box next to James to create a task and fill in some text. You will now have a task in your task list.



3. Complete a task by clicking **Complete**. The task will no longer be available in your task list – but don't do that yet! You can do a variety of things with a task, such as give it a due date or assign it to someone else. James also has some hints on adding reference documents, comments, and involving other people.



When you involve someone else in a task, it will appear in their tasks list. This enables them to contribute to the task such as add comments, documents, and even involve more people. However, only the person who is assigned the task with can actually complete it. In the following example we've added a document, a comment, and involved a person.



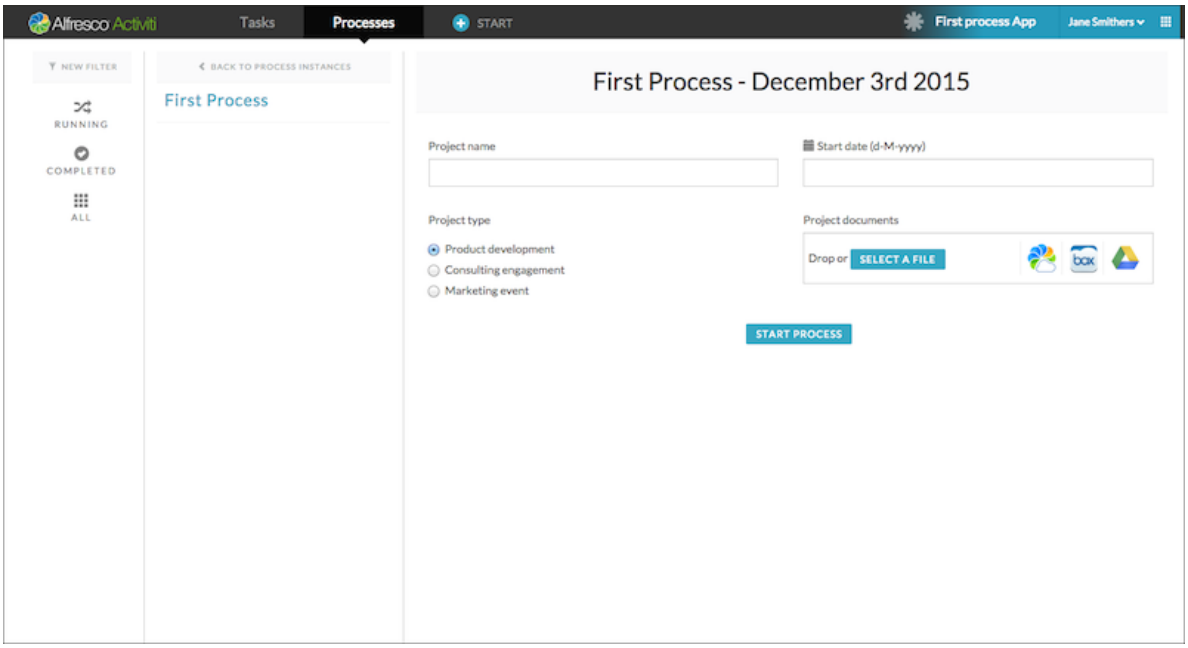
4. Click **Complete**. If you wish to view that task again, you can click the **Completed Tasks** filter on the left pane. By default, you will see all tasks you are involved with, however you can customize your view to:

    - Tasks that are directly assigned to you
    - Tasks where you are listed as a candidate
    - Tasks that belong to the group you're member of

    Note: Not all user accounts may have groups assigned.

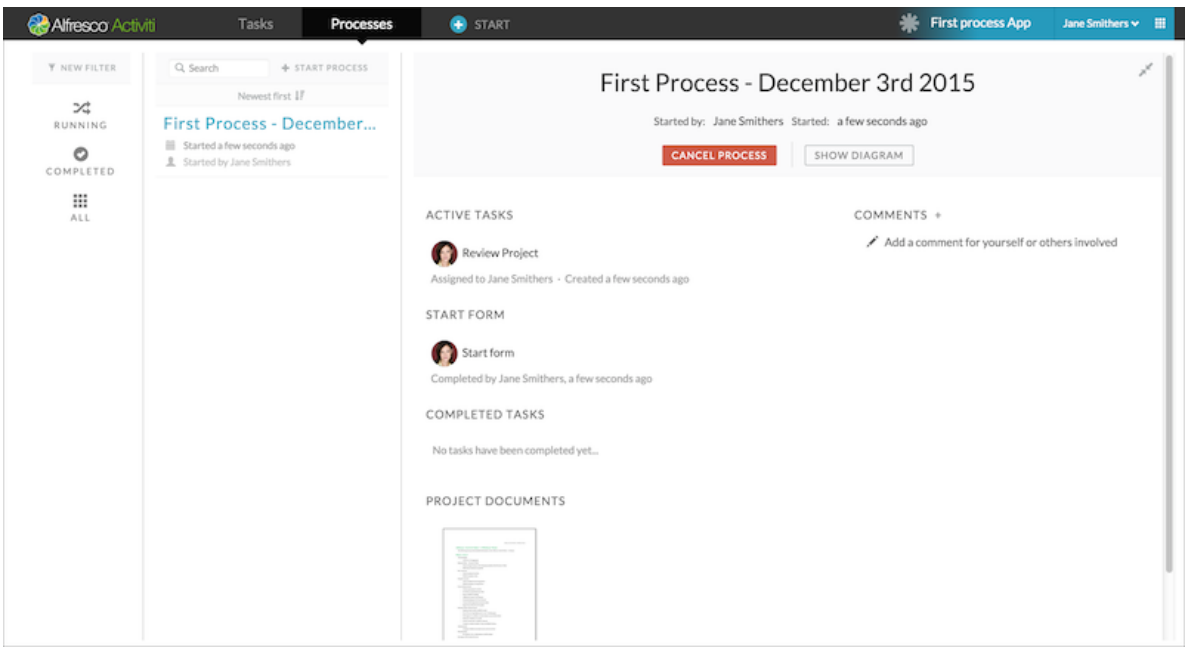
Now that the tasks have been created, let's start the process we designed earlier.

5. Click on the James hint box or **+ Start** in the header area. A list of available processes are displayed, which in our case will be only one. When you select it, the Start form we created above is displayed. You can also change the name by clicking the title on the right panel. By default the current date is added to the name of the process.

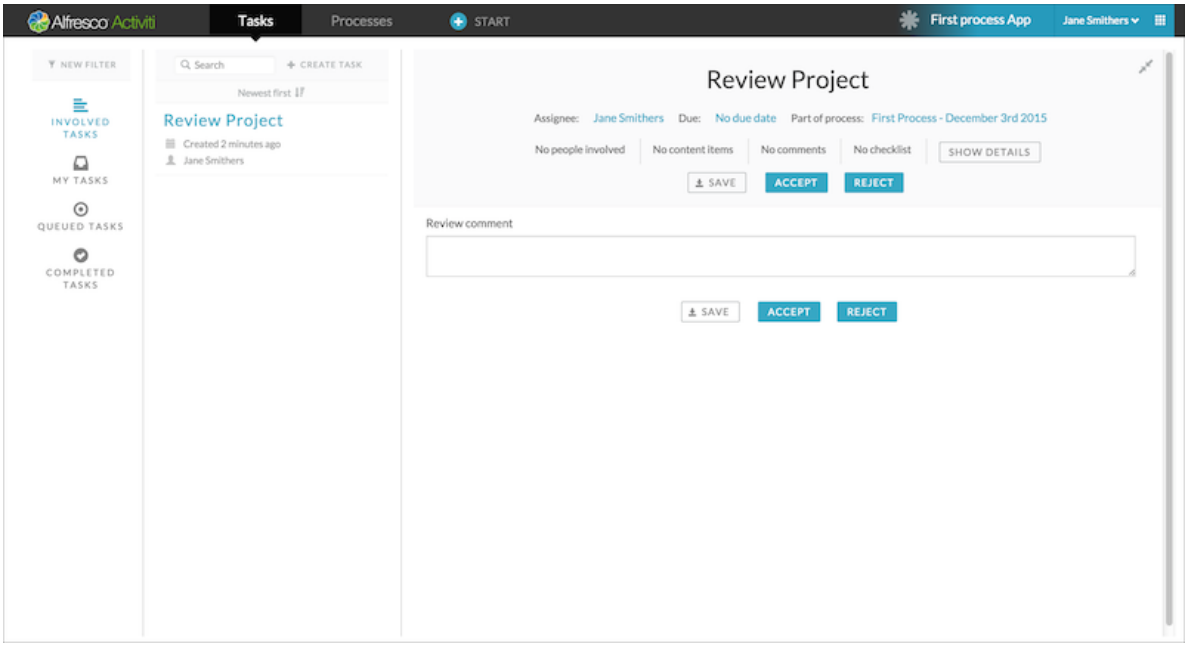


6. Fill in the form and click **Start Process**.

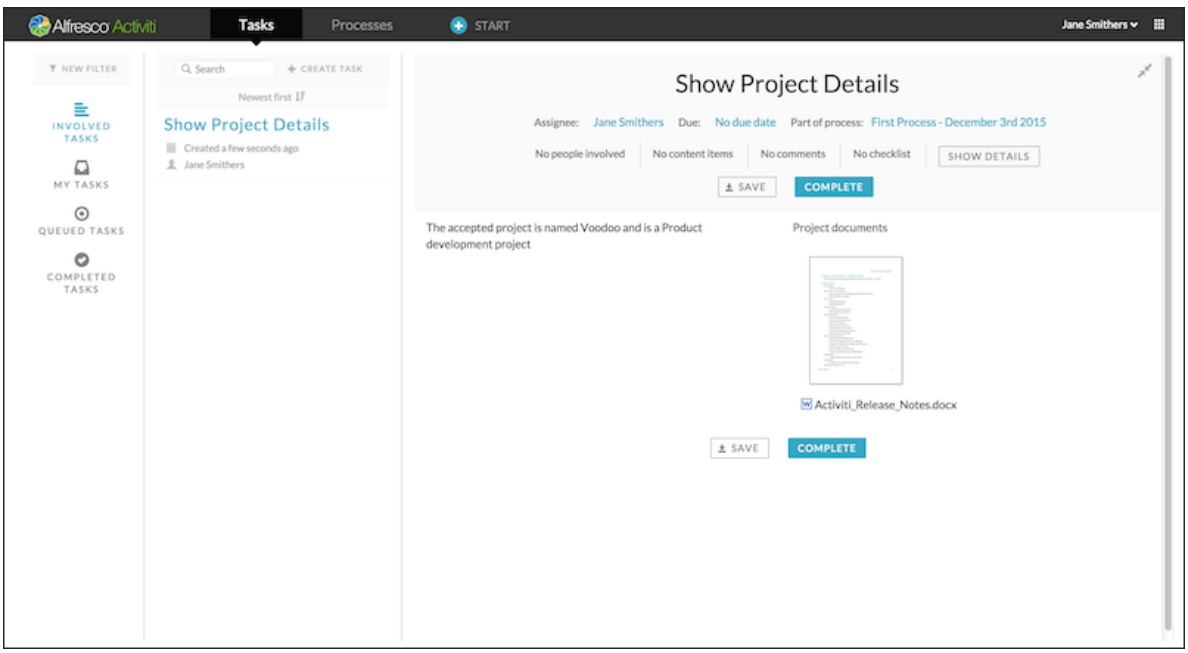
   You will be returned to the **Processes** page, showing the details of the newly started process in your process list.



You can always view a process to see what the current and completed tasks are, as well as add comments that will be available for anyone involved in the process at any stage. If you go to the **Task** page that we just created, you will see the first step in the process is that of a task to review the project, and accept or reject it. The task was assigned to you because it was set to the process initiator, and you started the process.



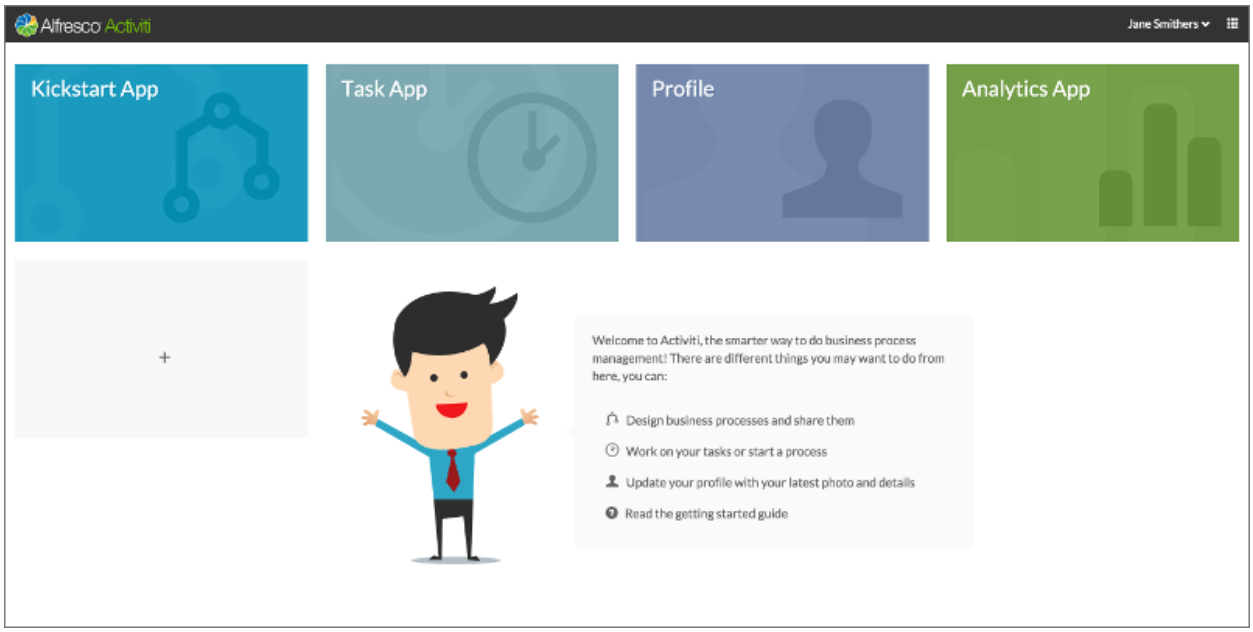
Before you fill in the review summary and choose accept or reject, you can still add people, documents, and comments by clicking on the **Show details** button in the task header area. You can get back to the form from there by clicking the **Show form** button. If you click the **Accept** button, the **Review Project** task will disappear and instead a new task, **Update Project List** will appear. This is because you defined it as the next choice step in the Step Editor, if the choice was to accept the project. You can just click the **Complete** button to move to the next step, which is a task to show the details of the accepted project.



When you complete this task, your task list and your process list will be empty. If you prefer to see all your tasks and processes in one place rather than through different process apps, you can use the **My Tasks** tile to get your complete task and process lists.

2. Alfresco Activiti

The Alfresco Activiti Web application is your user interface to Activiti. When you first log in to Activiti, you will see your landing page.



Each tile gives you tools for distinct sets of tasks.

You can get back to your landing page at any time by clicking on the Alfresco Activiti logo on the header.

Your landing page is dynamic, and new tiles will appear when you create new process apps in the Kickstart App and deploy them in the Task App.

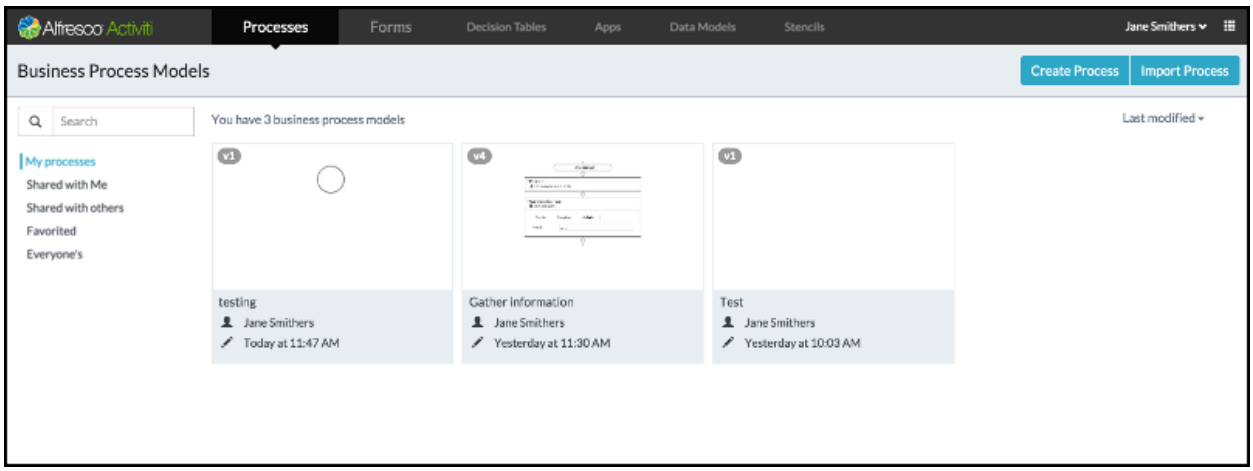
Your Activiti buddy James appears below the tiles to help you with a list of shortcuts for tasks you might want to do next. He appears in many places in the app to guide you, in particular when you haven't worked with the tasks on a page. He provides useful hints to help you design and share business processes, work with your tasks and processes, update your profile, and a link to the Getting Started Guide. The Getting Started Guide is the tutorial embedded in the product that will help you learn the basics of working with Alfresco Activiti.

Note	If you are an administrator, your landing page is slightly different. Instead of the Profile management tile, you'll find a more powerful set of tools called the Identity management tile.

All pages display the App Navigator icon in the far-right corner of the header. It provides useful 1-click shortcuts to various parts of the app. You can navigate instantly to all your process models, tasks, processes, stencils, forms, decision tables, quickly start any process, view the tasks and processes for a published and deployed app, or view and change your profile. As you deploy process apps, the App Navigator will also show shortcuts for the newly created process apps.

2.1. Kickstart app

Use the Kickstart app to create process models, forms, app definitions, and share your models and definitions with others. As you create items, they appear as tiles on their respective page. The Last Modified drop-down on the top-right enables you to sort the display order ranging from last modified, oldest first, name order, or reverse name order. Use the filter on the left to filter the list of displayed items. Additionally, if you are unable to find a specific process, use the search box to find more processes. If your processes require human input, then you will need forms to gather it.



You can filter the list of Business Process Models using the following options on the left:

- **My items** - View all your processes / app definitions / data models / stencils / reusable forms / reusable decision tables. The filter name changes based on the tab you are in. For example, in case of the Forms tab, it changes to My reusable forms and to My App definitions when you are in the Apps tab.
- **Shared with Me** - View items shared by others with you.
- **Shared with Others** - View items that you have shared with others.
- **Favorited** - View your favorite items.
- **Everyone's** - View all processes regardless of who created them.

Note: The **Everyone's** filter is applicable for admin purposes only. You can't use this option to allow others to reuse the model. To allow someone else to use this model, it has to be shared first.

The Kickstart panel includes the following tabs:

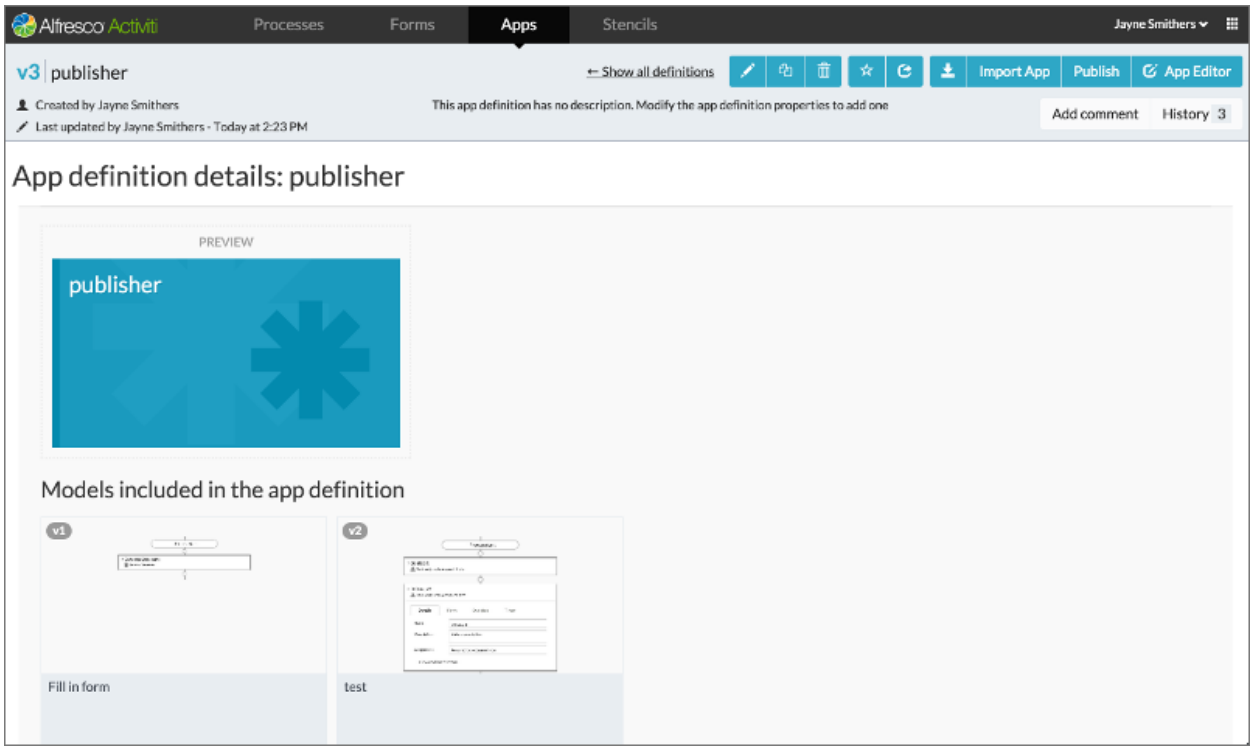
- **Processes** - Provide tools for creating new processes, modifying existing processes, and importing processes from outside Alfresco Activiti.

Note	If you haven't created any processes yet, then James will appear with shortcuts for creating a process. You can use the simple Step editor, or the more powerful BPMN editor. If you are not familiar with the BPMN 2.0 Business Process Model language, then the Step Editor is for you. However, if you'd like to create complex processes, then the BPMN Editor will let you use the full power of the language. It's helpful if you're familiar with BPMN 2.0 for using the BPMN Editor.

- **Forms** - Provide tools for creating new forms, and modifying existing forms. Filter the list of displayed forms using the options on the left. You can view all your forms, or just those shared by others with you, or those you have shared with others, or just those you have favorited. If you haven't created any forms yet, then a new button called Create a new form now! will appear on the **Forms** tab.
- **Decision Tables** - List decision tables that can be used across processes. Decision tables are an easy way to define business rules.
- **Apps** - Create new apps, modify existing apps, and import apps from outside Alfresco Activiti. You create an app to group one or more of your processes, so you manipulate them as one unit. You can make an app available for yourself and share it with others. An app can contain no process at all, which allows you to create simple task list.
- **Data Models** - Enable you to map your business data with a relational database or a custom API such as a customer database, patient database, and so on. You can create business objects to connect to an external database that can be accessed by all processes in your application.
- **Stencils** - A stencil is a customized process palette that is common to the Step editor, BPMN editor, and Forms editor. When you create a process or a form, you can specify a specific stencil or use the default for the editor you are using.

2.1.1. Kickstart editor

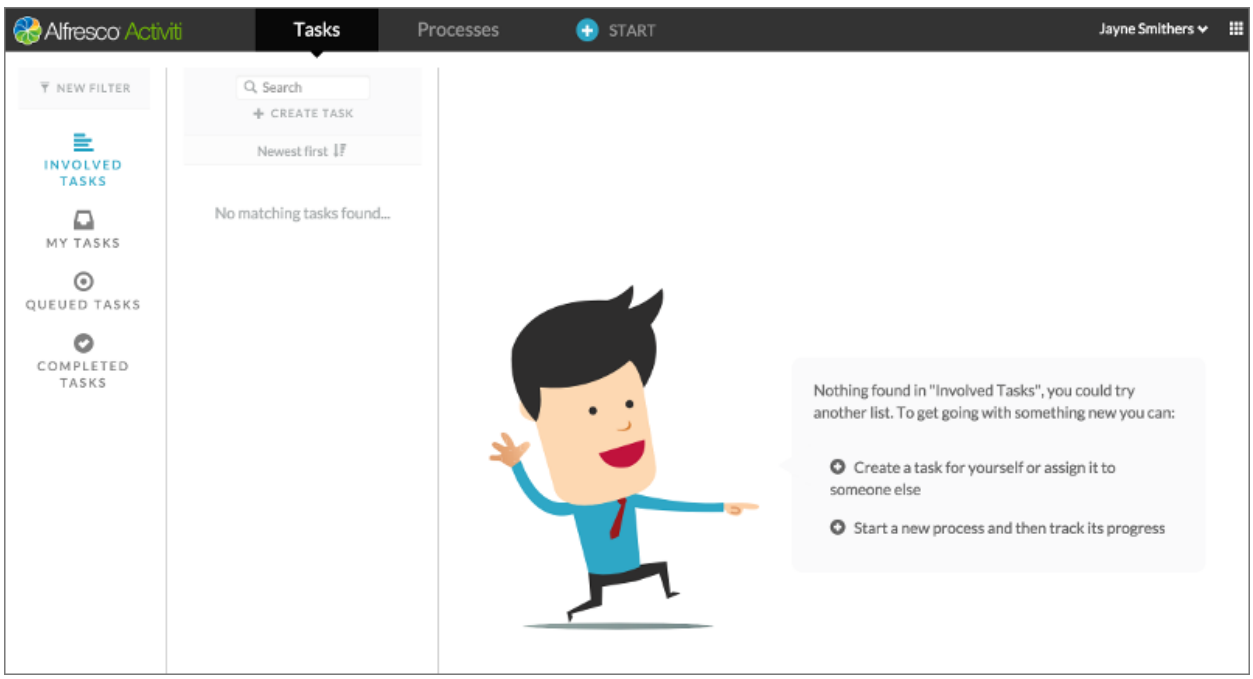
You can open the Kickstart editor by clicking a process definition, reusable form, reusable decision table, app definition, data models, or the stencils tab. The Kickstart editor provides features such as copy, comment, delete, add to favorites, share with others, and export. You can also open the corresponding editor to make changes to the content, and perform actions specific to the item type. For example, you can publish an app definition or edit a process.



In the above example, the Kickstart editor was opened for an app definition called publisher. The editor always displays the details of the selected item on the top panel along with a set of buttons on the top right. The right-most button opens the editor corresponding to the item displayed. So in the example, the right-most button opens the app editor. If a process definition created via the step editor is opened in the Kickstart editor, then the App Editor would open the step editor.

2.2. Task App

Use the Task App to access your task list and work on tasks assigned to you from the Processes tab. This is also where you initiate new processes and tasks.



The Task App menu bar has tabs for working with tasks, processes, reports, and a Start button, which is a shortcut to start a process using a published process definition.

Note	If you haven't created any tasks for yourself, and there are no tasks assigned for you from current processes or from other users, then James will appear with shortcuts to help you create a task or start a published process.

2.2.1. Tasks tab

The Tasks tab is organized into three columns.

- The left column lets you filter the list of displayed tasks. There are four pre-defined filters and a New Filter control which lets you define and name your own filters. Any filters you create are added to the list of displayed filters. See Using the new filter in tasks & processes.
- The middle column provides tools for creating new tasks, and lists the tasks included by the current active filter. Click on the accordion icon above the list of tasks to change the default display order from Newest first to oldest first, Due last order, or Due first order.
- The right column is displayed when you click on a task in the middle column. It displays the selected task details and also tools for completing open tasks and for viewing the audit log of a completed task.

Note: The Audit log button is only available for a completed process instance or a completed task.

Using the new filter in tasks & processes

When you create a new filter in the Tasks tab or Processes tab, you can filter by process definition, the state of the task/process, by task name, and by assignment. You can also change the default sort order.

Process definition
: Select an active (running) process name, and display only those tasks that are associated with that process.

State
: Choose to display tasks or processes based on its state. For tasks, select Completed or Open. Completed is selected by default. For processes, select Running, complete, or All. Running is selected by default.

Assignment

Select tasks in which you are involved, or tasks that have been assigned to you, or tasks where you are one of the several candidates. This is only applicable to the Tasks tab.

Sort
Sort the list by Newest first, Oldest first, Due last, or Due first.

Task name / Process Name
Type a string to search for matching task names or process name depending on the tab you're in.

Filter icon & name
Select an icon for your new filter by clicking on the funnel icon, and specify a name for the filter.

Note	If you have no tasks or processes running, then James will appear with a shortcut to let you create a new task for yourself or start an existing process and track its progress.

2.2.2. Processes tab

Use the Processes tab to start a new process from a list of published process definitions. The Processes tab is organized into three columns similar to the Tasks tab except that instead of tasks, process details are displayed. You can also create a new filter to filter by process definitions, process state, and by process name. See Using the new filter in tasks & processes for more details.

2.2.3. Reports tab

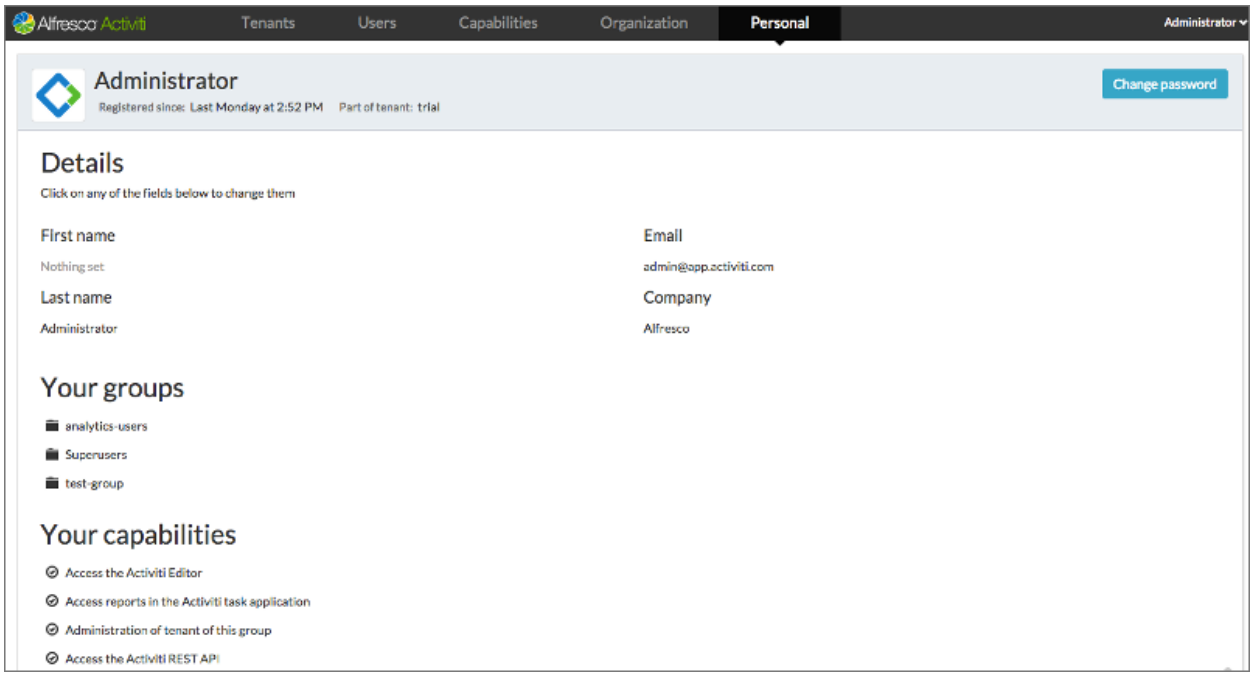
Use the Reports tab to generate reports based on the available parameters. You can view the reports that you saved in the Analytics App. For more information, see Analytics App.

2.3. Profile management

This tile will appear for you only if you are a user. This is where you manage your personal information.

2.4. Identity management

This tile is only available if you are an administrator. This is where you manage your personal information, users and groups in your organization, and tenants in your Alfresco Activiti engine.



The Identity Management tile presents tabs for managing tenants, users, capabilities, organization, and personal information. By default, the Personal information tab is displayed.

Note	The trial version of Alfresco Activiti has only one named tenant called trial.

2.4.1. Tenants tab

Use the Tenants tab for creating new tenants, and modifying existing tenants.

By default, the details of the currently selected tenant are displayed. You can edit the name of the current tenant and configure various settings as follows:

- **Logo** - Add or update your existing logo.
- **Events** - A log of management events for the tenant.
- **Alfresco repositories** - Configure your on-premise Alfresco repositories. See *Create Alfresco repository*.
- **Endpoints** - Configure your RESTful endpoints and Basic Authentication for endpoints.
- **Data sources** - Register your data sources for using in Data Model.
- **Document templates** - Upload a Microsoft Word (.docx) file that can be used as a template in processes.

- **Email templates** - Create new custom email templates, view or edit the existing templates (both standard and custom). For information on creating custom templates, see Custom email templates.
- **Config** - Configure settings for Box metadata support, validate decision table expressions, and enable or disable the option for involved users to edit forms. You can also define the minimum length for the password, and the date format for forms (for example, *D-M-YYYY*).

Custom email templates

Use Custom email templates to create your own set of templates in Activiti. You can use a custom template when creating human tasks in your process. This is particularly useful if you want to send a customized email notification as part of your process. You can also include the values from process variables and links to specific form outcomes.

Prerequisites:

- In IDM, create a user and assign the **Administration of tenant of this group** capability.
- Make sure to create users with valid email addresses as that will be needed for sending email notifications.

To create a new custom email template:

1. Go to the **Identity Management** app, click **Tenants** > **Email templates**. The email template page appears.
2. From the **Custom email templates** tab, click **Create new email template**. The Create new email template dialog appears.
3. Type a Name, Subject, and Email content for your template, and then click **Save**. You can also add variables such as *${taskName}* and *${taskDirectUrl}* as email content. The new template appears on the Custom email templates list.

You can edit and delete an existing custom template by selecting the edit (pencil) and delete (bin) icons in the Email Templates respectively. Once an email template is created, you can search it by entering a string for matching email templates.

Alternatively, you can also create a custom email template within the Kickstart editor when adding or editing a human step in a process.

To create a custom email template via the Kickstart Editor:

1. Go to **Kickstart App** > **Processes**, and create a new process with a human task or edit a human task in an existing process. Make sure to create a form or reference an existing form with the step.
2. In the Human step, check **Allow email notifications** and then select **Custom template**.
3. Type a Subject, and Email content for your template, and then click **Save**. A new custom email is created.

Note: Ensure that you don't give an existing name to your new template, otherwise an error *Name already taken* is displayed.

To use a custom email template in the process:

1. Go to **Kickstart App** > **Processes**, and create a process with at least two human steps.
2. In the **Human step** dialog > **Details** tab, check **Allow email notifications**.
3. In **Email**, select **Email template**, and then the custom template that you created in the above section (Create a new custom email template section).

To complete a task using custom emails:

1. Create a new app for the email process that you created above and then publish it.
2. Access the newly deployed App from the main Activiti page.
3. Start a new process and access your tasks.
4. Click **Complete** to complete the required tasks.

Your task is completed and custom emails are sent to the assigned user of the task.

2.4.2. Users tab

Users tab provides tools for managing users. The current users are displayed on the right panel. You can select from the list of users and use **Select an action** to change details, status, account type, password, and primary group of the user.

You can also create a new user, or filter the list of current users by status, account type, email or name, and company.

2.4.3. Capabilities tab

Use the Capabilities tab for managing the capabilities and groups of users that are available for this tenant. There are two types of groups:

- **Capability groups** - Groups that can be granted with variety of capabilities.
- **Organization groups** - Functional groups that reflect the structure of your orgnization.

The following capability groups are available by default:

- **Analytics-users** - Access Analytics app to view reports in the Activiti analytics app.
- **Superusers** - Administration of tenant of this group gives full administration rights for the current tenant to the selected group.
- **Kickstart-users** - Access to Kicstart app that allows the you to design and publish process definitions.

An Administrator can grant the following capabilities to any of the capabilities groups:

- Access Analytics app
- Access Kickstart app
- Access the Activiti REST API
- Access to all tenants' models
- Administration of tenant of this group
- Publish app to user dashboard
- Upload license

You create and delete capabilities groups, add and remove users to and from a group, and add and remove capabilities to and from all users in a group.

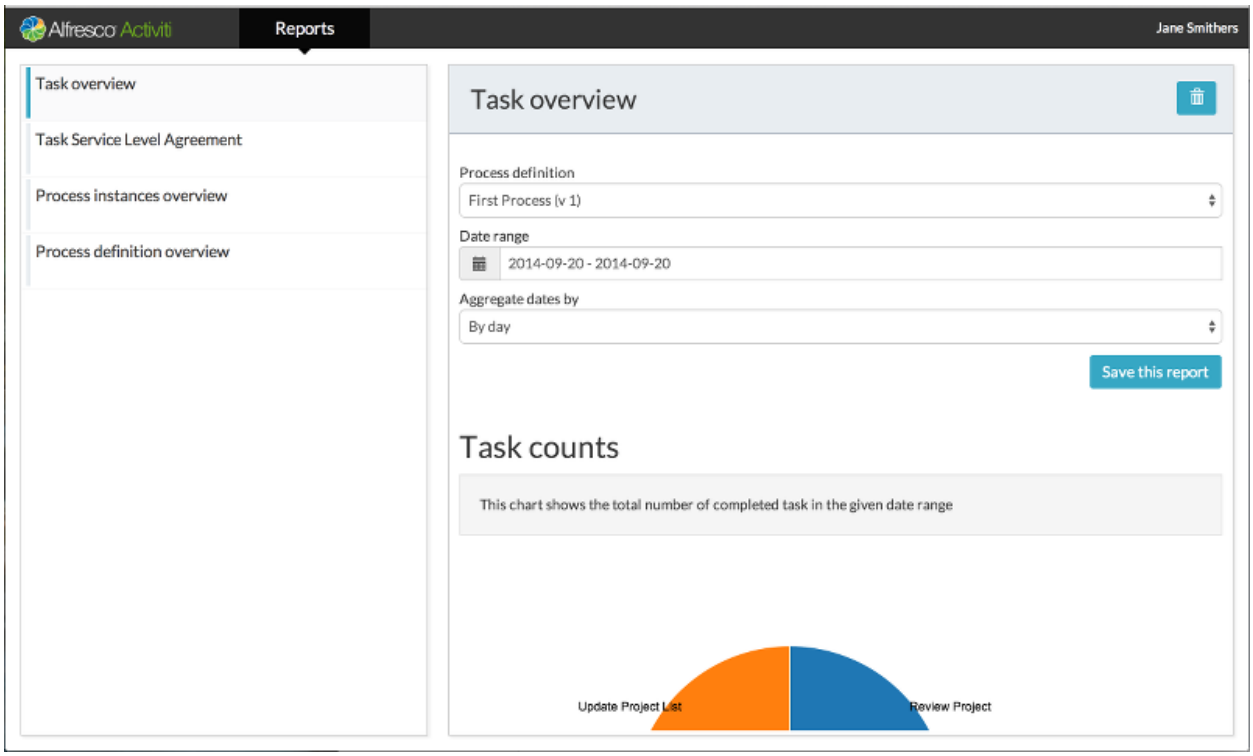
2.4.4. Organization tab

Use the Organization tab to create functional groups that reflect the structure of your organization. You can also add and remove users to and from a group, and create subgroups within this kind of group.

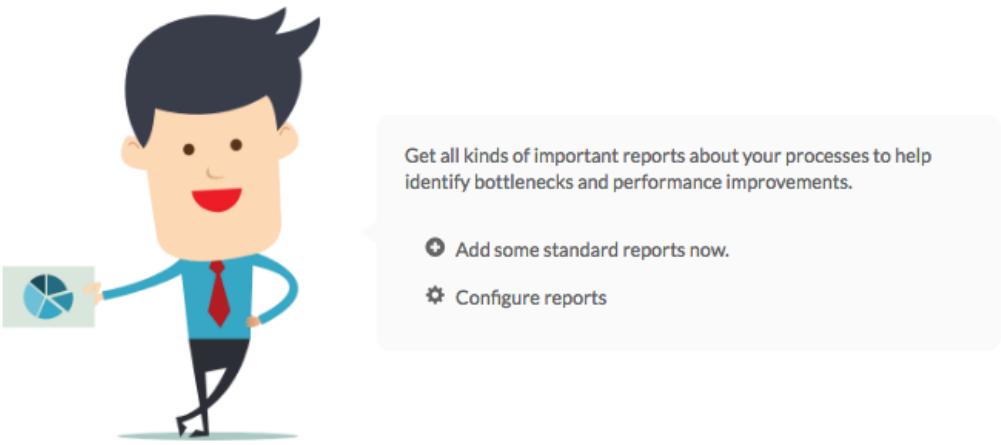
Note	When you remove a group, it will be deactivated, until all its tasks are complete. To remove the group completely, click the remove button a second time. When you remove a group, this will remove all its sub groups too.

2.5. Analytics App

Use the Analytics App tile to add standard reports and configure custom reports for performance and throughput statistics of your processes. You can view the Analytics App tile only if your account has the Analytics capability. Before generating process reports, make sure to run your processes in Activiti at least a few times.



When you visit the Analytics app for the first time, your Activiti buddy James appears on the welcome screen with useful hints.



The Analytics app has the following tabs:

- **Reports** - Use this to add standard reports in Activiti and view the existing reports.
- **Configure** - Use this to configure standard reports and custom reports.

2.5.1. Configuring standard reports

In Activiti, you can add Standard reports at a click of a button. You can choose to add all standard reports at once or configure only the reports you're interested in. For example, you can configure your report panel to isolate Task related reports such as Task overview and Task service level agreement reports, or custom reports that are based on generated reports (see Customizing reports).

To add standard reports:

- From the **Analytics app** > **Reports** tab, click **Add some standard reports now** link. The following standard reports appear in your Reports panel on the left:
    - Process definition heat map
    - Process definition overview
    - Process instances overview
    - Task overview
    - Task service level agreement

Alternatively, you can also add the same set of standard reports via the Configure tab. To remove your existing reports from the Reports panel, click **Reset all my reports**.

Once you have added the standard reports, you can access them from the Reports panel and generate them based on the required filter parameters. If the data is available, it will be presented in graph and tabular form, depending on the report selected.

2.5.2. Filtering reports

You can filter most reports by the following parameters:

- Date range
- Process defintion
- Process Status
- Task (Task related report only)
- Task Status (Task related report only)

Some reports such as Task service level agreement and Process instances overview reports have additional parameters.

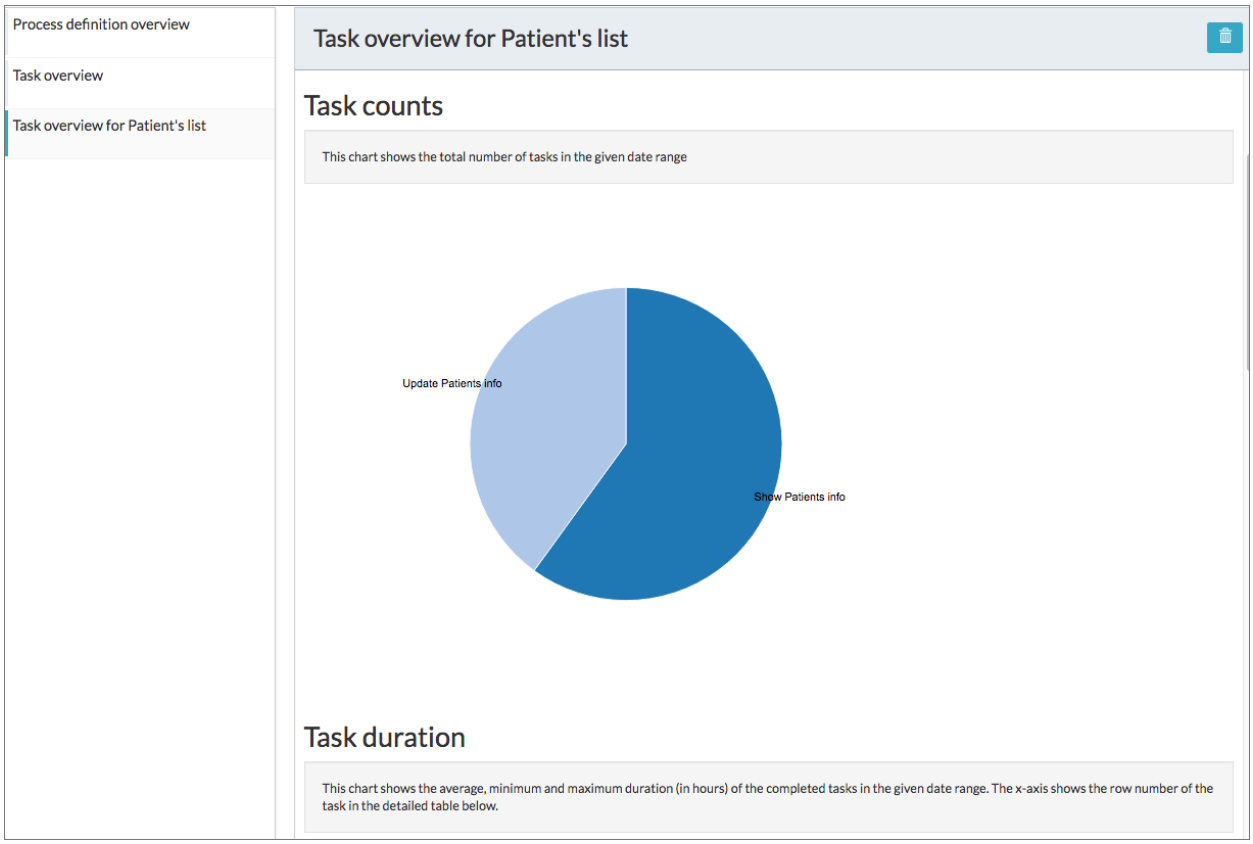
2.5.3. Customizing reports

You can customize reports by selecting the Process Status and Date Range parameters. You can also create new reports by modifying the filter option of an existing report and saving it with a new name.

To generate and save a Task overview report:

1. Sign in to Activiti as a user with Administrator privileges.
2. Click **Analytics App** > **Configure** and then Task overview.
3. Select from the following filter options:
    - **Process Definition** - Process definitions for the selected user.
    - **Date Range** - Tasks from Today, Yesterday, Last 7 days, Previous month, Current year, or Custom Range.
    - **Task Status** - All tasks, Active, or Complete.
    - **Aggregate dates by** - Tasks by hour, day, week, month, or year.

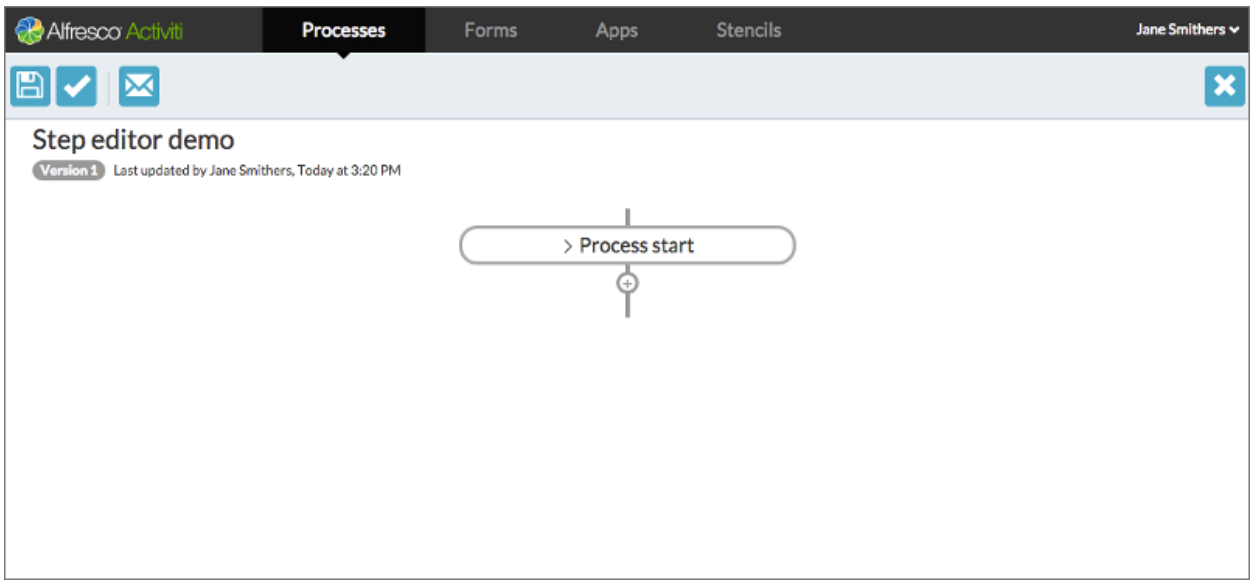
      Relevant data for Task Counts, Task counts by assignee, Number of tasks divided by date interval, Task Duration, and statistics of all tasks are presented in graphical, tabular, and table formats. There is also an option to view the previous chart data in a table format.
4. Click **Export Data** to generate the report in csv format.
5. Optionally, to save the report with the selected filter options, click **Save this report**. You can also choose to save the report by a new name for easy identification. For example, if your report is specific to a task called Patients List, you could save the report as Task overview for Patients' list.



You can generate all other reports in the same way by using the appropriate filter options. You are now ready to explore the advanced reporting and analytic features in Activiti.

2.6. Step editor

The Step Editor guides you through creating a business process through a sequence of simple steps. The processes you create using the step editor do not exploit the full power of BPMN 2.0 like those created by the BPMN editor, but you can use it to design both simple and quite complex process models, without knowledge of BPMN 2.0.



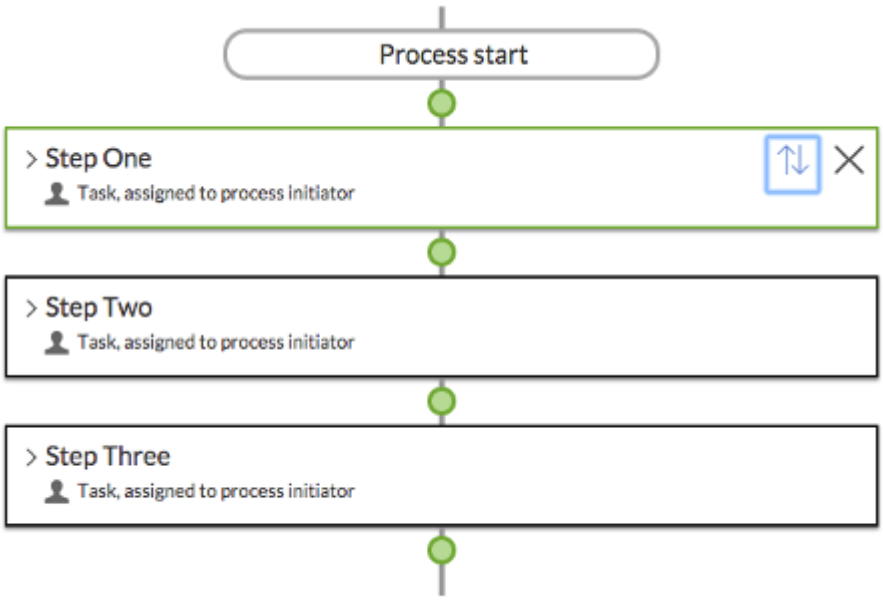
The editor has a menu bar with buttons to save your model, validate that the model is a complete BPMN 2.0 model definition, provide feedback to the Alfresco Activiti team, and to close the editor.

When you open the step editor on a new process definition, you can see the first step, the Process start step is already added to the process diagram for you. When you mouse-over a step, the stop becomes click-able. Click on it, and the details of the step are displayed and can be edited. This design principle is reflected throughout the Alfresco Activiti app. You can mouse-over and click text areas to modify their content, and variables to change their values. So for the Process start step, you can click on the single Process trigger variable and choose the trigger type:

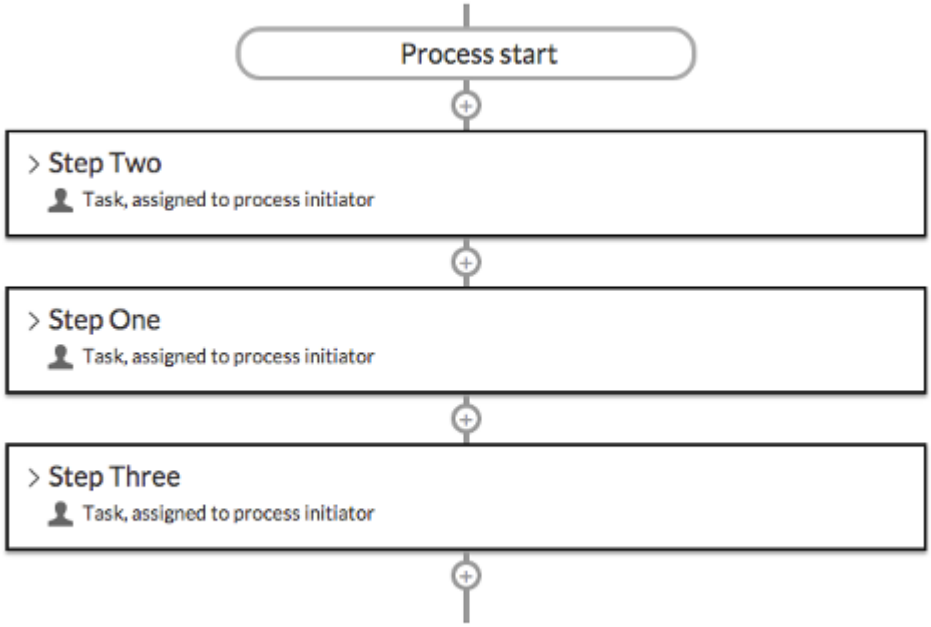
The editor will guide you in creating your process. For example, when a form is required, it will present you with a list of existing forms and provide you with a button to create a new form. So for example if you choose the Started by User filling in form option for the Process trigger variable in the Process start step, you would see the following dialog:

Below the last step in a sequence, there is a + (plus) icon. Click on this to add a step to your process.

You can move steps around in your process Click in the top-right of the step and the step will be outlined in green, and the + icons will change to green discs.



Click the green disk at which you want your highlighted step to move, and the step is moved to that position in the flow:



In addition to the Process start step, there are five types of step you can add to your process.

2.6.1. Human step

A human step is a task to be completed by a user. You choose who to assign the task to, provide a form for that user to complete, define a due date for the task, and set a timer. If a timer is triggered, it will allow Alfresco Activiti to take an action related to the task, such as reassign it to another user and so on.

The Human step dialog is divided into four tabs:

- Details tab
- Form tab
- Due date tab
- Timer tab

Details tab

Property	Description
Id	A unique identifier for this element
Name	A name for the task.
Documentation	A description of the task.

Property	Description
Assignment	Configure to who this task should be assigned. You can assign the task to one of the following assignees: ***Assigned to process initiator*** The user that started the process instance, which could be you, or a user you have shared the process definition with. The process initiator is the default assignee. ***Assigned to process initiator's (primary) group manager*** The group manager of the user that started the process instance. ***Assigned to single user*** When selected, an additional Assignee field is displayed enabling you to search for a single user or select someone using an email address. If that person is not currently an Alfresco Activiti user, they will receive an invite. ***Assigned to group manager*** When selected, an additional Group field is displayed enabling you to search for a group manager or select a form field (providing you have defined a form). Only users that have a primary group defined will have a group manager. You can define a primary group via **Identity Management** > **Users** > **Select an action** > **Change primary group**. ***Candidate users*** When selected, an additional Candidates field is displayed enabling you to add one or more candidates. You can add Alfresco Activiti users or select someone using an email address. If that person is not currently an Alfresco Activiti user, they will receive an invite. All of the selected candidates are eligible to complete the task. The task will show up in their *Queued tasks* task list. The task is not assigned until they have claimed it, which will make the user the assignee. ***Candidate groups*** When selected, an addtional Groups field is displayed enabling you to add one or more groups of Alfresco Activiti users. The task will show up in their *Queued tasks* task list. The task is not assigned until they've claimed it. The other users won't see that task in a task list anymore.

Property	Description
	Allow process intitiator to complete a task When checked, the user that started the process instance (process initiator) can complete the task. This is checked by default. This option is available only for Candidate Groups, Candidate Users, and Assign to single user options.

Form tab

You can select a form to display when the task runs. You can select an existing form, or create a new one. Forms that you create here while designing your process definition are accessible to steps in this process definition only. Forms that you have designed in the Forms tab of the Alfresco Activiti app can be reused by any process definition owned by someone you have shared the form with. Both types of form are listed in the chooser dialog. You can filter the available list of forms by entering text in the Filter box.

Due date tab

If you specify a Due date, then the time remaining until that date will be displayed in the task details when the process is running. If the task is not completed in that time, then the amount of time since the due date is displayed. You have the following options for setting a due date:

No due date for this task
: This is the default value.

Fixed duration after creation
: Specifies a Due date in years, months, days, hours, minutes and seconds after the task is started.

Based on field
: Select a date field from a list of those available in forms of this process definition. You can add or subtract a specified amount of time in years, months, days, hours, minutes and seconds from the value of the chosen date field to create a Due date.

Based on variable
: Select a variable from the the list of those available in forms of this process. You can add or subtract a specified amount of time in years, months, days, hours, minutes and seconds from the value of the chosen date field to create a Due date.

Timer tab

Timer is similar to Due date, except you specify a time after which some action will be performed on the task by Alfresco Activiti. You can also specify an action for the task to be taken when the timer completes.

You have three options for setting a timer:

No action
: This is the default value.

Reassign task
: You specify another assignee in exactly the same way as you specify the original assignee on the Details tab. When the timer completes, the task is assigned to the specified user, candidates users, or candidate groups.

Keep task
: When you specify Keep task, a new Timer date reached substep appears inside the current step with the + icon underneath it. You can add one or more subtasks inside this step by clicking this icon. When the timer completes, the task remains active, and the first substep becomes active too. The process continues running substeps as each substep is completed. Note that when you specify substeps here, the list of steps available now includes a **Go to** step. This allows you to choose one of the main process steps to run after this one.

End task
: When you specify End task, a new Timer date reached substep appears inside the current step with the + icon underneath it. You can add one or more subtasks inside this step by clicking this icon. When the timer completes, the task ends, and the first substep becomes active. The process continues running substeps as each substep is completed. Note that when you specify substeps here, the list of steps available now includes a Goto step. This allows you to choose one of the main process steps to run after this one.

End the process
: When the timer completes, all active tasks in the process are canceled and the process ends.

2.6.2. Email step

When an email step starts in a running process, it sends an email with a fixed text body and a fixed title to a single or multiple recipients.

The email step dialog contains two tabs that let you fully define the task.

Name and Description are simple text fields that help you and others to identify the task in your task list.

Recipient type lets you choose who receives the email defined in this step:

Process initiator recipient
The user who starts the process is the sole recipient of the email. This is the default.

Single user recipient
If you choose this option, a Recipient field is displayed to allow you to search for single user or select someone using an email address.

Multiple user recipients
If you choose this option a second Recipients field is displayed to allow you add one or more users. You can add Alfresco Activiti users or select someone using an email address.

2.6.3. Choice step

A choice step enables you to start one of two or more sequences of substeps for your process, based on conditions.

Use the Name and Description fields in the choice step dialog to define the task for your task list.

When you select the Choices tab for a new choice step, it shows two choice boxes. You can use the + (plus) icon between them to add more choices. Click the choice box you to edit the choice and name it. You can also add from one of the following conditions:

No condition
This choice runs its sub-steps if none of the other choices conditions are met. Note that only one of the choices in a choice step can specify this condition for the model to validate. This is the default.

Form field
This choice runs its sub-steps if the value of a field in a form satisfies a conditional statement. If you click this option, the following options are available:

- Select a field in a form that is used in this process definition.
- Choose an operator from equal, not equal, less than, greater than, less than or equal to, greater than or equal to, empty, not empty.
- Specify a value. For example, select a radio button field named **direction** from a form, choose the **equals** operator, and type the value **Left**.

Form outcome
This choice runs its substeps if the outcome of a form that matches the one specified for the choice is selected by the person assigned with the task. If you click this option, the following options are available:

- Select an outcome of a form used in this process definition.
- Choose an operator from equals or Not equals.
- Select a value of the outcome from the list. For example, select an outcome named **direction** from a form, choose the Equals operator, and choose the value **Turn left** from the drop-down list.

There are two steps that you can add at the end of a substep sequence in a choice step that change the flow of control in the process.

End process Step

An end process step is available only when defining a substep within a choice step. You use an end process step to stop the process within a choice step in your process definition. Since this is a terminal step, no + (plus) icon appears after the step.

In the **End process step** dialog > **Details** tab, define the task name and description.

Goto step

The Goto step is available only when defining a substep within a choice step. You use a goto step to jump to a named step within your process definition. Like the End process step, it is a terminal step and no + (plus) icon appears after it.

1. In the **Goto step** dialog > **Details** tab, type a Name and Description in order to help you and others to identify the tasks in your task list.
2. Select a Goto step in this process definition to follow next.

   The process definition used here illustrates models for driving a car. If you turn left, then you continue your journey. As long as you continue turning left, your journey continues. If you turn right, you drive a short distance to your final destination. The goto step provides two ways of managing the flow of control in a process:

3. You can implement repetition, as illustrated.
4. You can also move the flow of tasks to another step in the current process.

2.6.4. Sub process Step

A sub process step enables you to create a step that itself contains a sequence of steps that constitute a complete process definition. When saved, this definition is added to the list of substeps available to your main process definition. This gives you a method of managing complex processes by refining repeated sequences of steps into a sub step. This can make your process definition easier to comprehend visually.

The sub step step dialog contains one tab that lets you fully define the task.

A sub process lets you choose a sub process that you have already defined in this process definition, or you can create a new sub process that is reusable in this process definition.

2.6.5. REST call

This step allows you make an arbitrary REST call. You can define a full endpoint directly or use an endpoint defined by an administrator on your Alfresco Activiti server. You can supply parameters to the call directly in the URL or from process variables in forms, and you can extract properties from the JSON response into process variables for use in your process definition.

Note	A user with administration privileges will need to add endpoints for standard REST calls, with Username and Password pairs that are permitted for basic authentication. An administrator can add these endpoints and authentications on the Tenant page of the Identity management app. The benefit of using standard endpoints is that they can be easily switched for test and deployment configurations. It is also possible to use a REST step to call Activiti's own REST API.

The REST call step dialog contains four tabs that let you fully define the call.

Name and Description are simple text fields that help you and others to identify the task in your task list.

You define the URL for your REST call in this tab.

HTTP Method
: This is the method associated with the REST call. The default is GET, but you must select between GET, POST, PUT, and DELETE based on the documentation for your chosen API call. The example shown in the screenshot, is using the `api/enterprise/app-version` REST call, which is documented as a GET call.

Base endpoint
: You select one from a list of endpoints that have been defined by your administrator. In the example the endpoint for the local Activiti server REST API, http://localhost:8080/activiti-app/, has been chosen.

Rest URL
: Copy the URL fragment from your selected REST API call. In this example we are using `api/enterprise/app-version`.

Form Field/Variables

You can insert values previously submitted in any form (or variables) in your process definition, into the REST URL. The value will be inserted at the position of the cursor in the Rest url field.

Some REST calls require a JSON request body. You can add one or more JSON properties using this tab.

For each property you define the name, property type and value. The value can either be a fixed value, or you can select the value of a form field from a list of available form fields in your process definition.

REST calls return a JSON response body. You can define one or more pairs JSON response properties and process variables. When the step completes, each process variable will contain the value of the returned response property. You can use those values later in your process. In this example, the returned JSON property edition will be contained in the process variable *activitiedition*, which is a form field in a form used for displaying the edition string later in the process definition.

2.6.6. Generate document

Use this step to generate a Microsoft Word or PDF document from a template in Microsoft Word. The process step will substitute any variables you place in the template document with process and form variables. You can upload global template documents for use by all users, or upload personal template documents for your own use.

Note	A user with administration privileges can upload global templates. An administrator can add templates on the Tenant page of the Identity management app.

The Generate Document step dialog contains the following tabs to define the task:

Details tab

- **Name and Description** - Type the name and description of your task.
- **Output name** - Type the name of your output document.
- **Output format** - Click the format that you want to view your generated document: PDF or Word.

Template tab

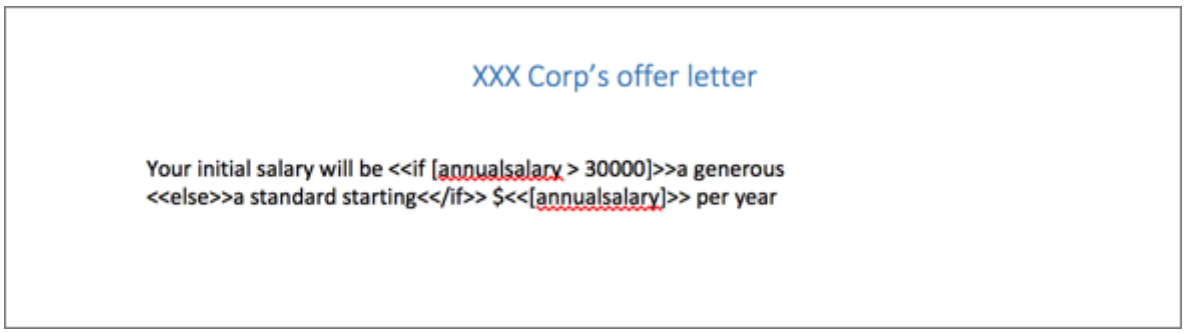
Select from a list of company templates that an administrator has uploaded or upload your own personal templates by clicking **Upload Template**. In the above example, the `offer.docx` company template is selected.

You can also filter the list of company templates with a search string, and download any template to see what form and process variable substitutions are made in the template.

Variable tab

Enter a variable name that you have used in the document.

In the template, you can substitute `<<[name]>>` in the output document with the form variable name, for example:



Templates are processed using the LINQ reporting engine. Refer to the programming guide in the link for the format of the templates, and examples to help you create your own templates.

You can also use expressions to build more complex templates. For example, the following excerpt was used in an HR offer letter of XXX Corp called *offer-letter.docx*:

```
    Your initial salary will be <<if [annualsalary > 30000]>>a generous
<<else>>a standard starting<</if>> $<<[annualsalary]>> per year
```

The sample template referred above uses conditional expressions that tests the value of the form variable `annualsalary` and outputs one of the two different text phrases, depending on that value.

To test the *offer.docx* template, create a process definition that uses the template. For example:

1. Create a process with the option started by user filling in a form.
2. Create a form called starter with four fields: a text field with the ID name, a set of radio buttons with the ID department, and two number fields with the IDs annualsalary and annualbonus.
3. Once you have filled the form, the Generate Document step will take the `offer.docx` template (mentioned above) and generate a document with a name defined by value of the Variable tab, `offer-letter.docx`.
4. Create an app to include the process definition that you just defined, and then publish the app.

5. Click **Start Process**. The Generate Document step is executed and the `offer-letter.docx` document is generated.

In the above example, the Generate Document step is the last step in the process definition, therefore you can view and download the generated document of the completed process in Activiti process view.

2.6.7. Decision step

The decision step enables you to create a Decision Table. A decision table is an easier expression to creating business rules.See the Business Rules - Decision Tables section for more details on Decision Tables.

2.6.8. Content-related steps

Use this section to link create content related steps.

Retrieve Alfresco Properties

The Retrieve Alfresco Properties option enables you to retrieve content-specific properties from Alfresco One and map it to a form field or variable, for example, properties of a document linked from Alfresco. You can retrieve document information after a document is added or referenced via the Attachment form field in Alfresco Connector.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Alfresco properties	Retrieves Alfresco properties for content stored in the form editor or variable, and allows mapping them.

Update Alfresco Properties

The Update Alfresco Properties option enables you to update content-specific properties in Alfresco via a form field or variable. For example, you can update properties of a document linked from Alfresco via a form attachment field, or process variable.

The Properties sheet displays the same fields as Retrieve Alfresco properties, except that is is used for updating properties rather than retrieving.

Call Alfresco Action

The Call Alfresco Action enables you to invoke the standard Alfresco One actions from Activiti.

Property	Description
Details tab	
Name	The name of the content-specific step.
Description	A description of this step.
Target tab	
Content	Retrieves Alfresco properties for content stored in the form editor or variable based on your selection.
Act as	Identity of the caller: Process initiator or Specific User. Selecting **Specific User** lets you select a different user.
Repository	Changes the repository account. For example: Alfresco One One, Alfresco in the Cloud.
Action tab	
Action	Lists a range of actions specific to Alfresco One. Select the options to make changes to the default name and value depending on your requirement. The options are as follows: ***extract-metadata*** Extracts embedded metadata from files and added to the file properties. Alfresco one supports Microsoft Office document properties, LibreOffice, and a number of other formats. ***move*** Moves the files and subfolders to the locations of your choices in Share if you edit the following value with the exact location of your document in Share: *workspace://SpacesStore/<ID>* ***add aspect*** Adds a property aspect to files for additional behaviours or properties.

Property	Description
	specialise-type Changes a file's content type, if applicable. For example, you can changes a standard file into a policy document and adds the appropriate metadata for that content type. *script* Runs a custom JavaScript script from the Data Dictionary/Scripts folder. There are a number of sample scripts available. The list can vary depending on how Alfresco One is configured. *check-in* Checks in files that are currently checked out. For example, files will be checked in before being moved to another folder. Select the option to indicate whether they will be checked in as minor or major versions. *transform and copy content* Action for transforming and copying content. You can add copies of files, in the format of your choice, to another location. For example, you can generate a copy of a Word document in PDF format in a different folder. *remove-features* Removes a property aspect from files to remove functionality or properties. *check-out* Checks out files automatically with a working copy created in the location of your choice. Select the option to associate a name or type with the file. *copy* Creates copies of files in the location of your choice. Set the additional deep-copy and overwrite-copy options to true if you want to copy or overwrite sub-folders and their contents. *transform-image* Action for transforming and copying image files in the format of your choice to another location. For example, you can generate a copy of GIF file in PNG format in a different folder.

Property	Description
Action Parameters	View or update parameters of the action selected in the previous field.

Publish to Alfresco

This step enables you to write a document or all documents uploaded in your process to an Alfresco repository. This can be an Alfresco One on-premise repository, or Alfresco in the Cloud.

Note	A user with administration privileges will need to add accounts for the Alfresco repositories that you can publish to. An administrator can add Alfresco repositories on the Tenant page of the Identity management app. The list of repositories you can publish to is then shown on your Personal Info page. If you click on a repository, an account to access the repository is added for you.

The Publish to Alfresco step dialog contains three tabs that let you fully define the task.

Name and Description are simple text fields that help you and others to identify the task in your task list.

Publish all content loaded in process
: this is the default. All files that have been uploaded in an upload field in a form before this step are published to the specified location in the repository

Publish content uploaded in field
: If you select this option a second field Form field displays a list of form fields from all the forms in your process. You can select one from the list.

Destination
: This is the folder in an Alfresco repository to which the selected content will be published. Click Select Folder to display a dialog that lets you choose a folder from the available Alfresco repositories defined in your Alfresco Activiti app. Once you have selected a folder, the Repository details and folder path are displayed in this field.

Subfolder

If you check create or reuse subfolder, a second field Based on field displays a list of fields from all the forms in your process. You can select one from the list. A folder with a name based on the content of the selected field will be created or reused within the specified destination folder to publish the content selected. If you do not select this option, all the items of content will be published directly to the specified destination folder.

Publish to Box

This is similar to the Publish to Alfresco step, but for Box. (https://www.box.com/).

Note that a Box account needs to be configured in the **Identity Management** > **Personal** tab.

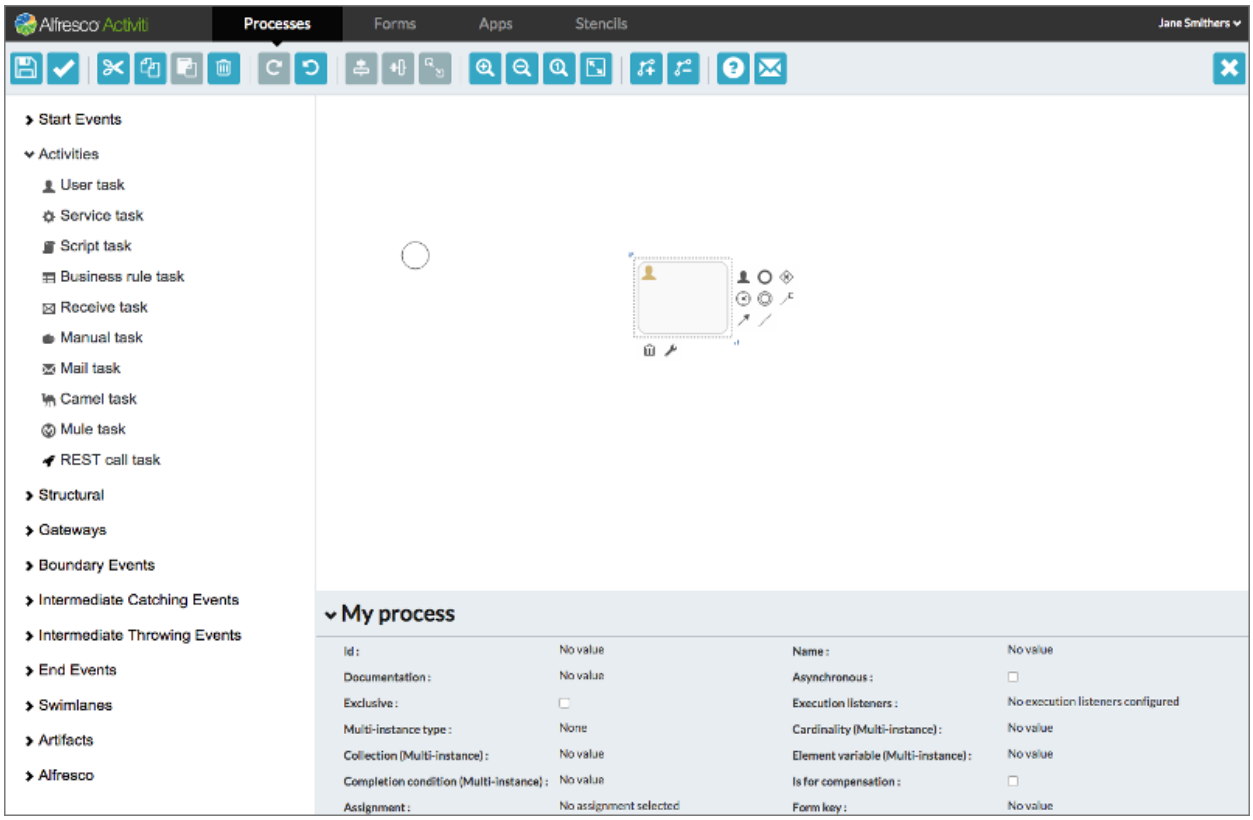
Publish to Google Drive

This is similar to the Publish to Alfresco task step, but for Google Drive. (https://www.google.com/drive/).

Note that a Google Drive account doesn't need to be configured like for Box or Alfresco. A popup shows up when you have to select a document/folder and no account is found. This popup will allow you to log in with the Google Drive credentials and use this account thereafter.

2.7. BPMN editor

With the BPMN editor you can create process definitions using the capabilities of BPMN 2.0. You build your process by dragging and dropping from a palette of grouped components to a canvas on which your process diagram is built.



The BPMN editor is structured into several areas:

Palette

On the left side of BPMN editor is the palette, which consists of collapse-able groups of BPMN objects.

Canvas

On the right side of BPMN editor is the canvas, where the BPMN objects can be added to create a process model.

Properties sheet

Below the canvas is the properties sheet, which shows the properties of the selected BPMN object on the canvas, or if no BPMN object is selected, the properties of the process itself. You can click on any of the properties to modify its value. The property sheet is collapse-able to allow you more screen space to view your process diagram.

Toolbar

The toolbar is displayed on the top with a set of grouped command icons. You can save and validate your model, delete selected elements in the diagram, cut, copy and paste selected elements, undo and redo the last action, zoom the process diagram, eliminate crossing connector lines by adding and removing bend-points, view the BPMN editor tour, and provide feedback to the Alfresco Activiti team.

When you first use the BPMN editor, a short guided tour runs showing you the components of the editor and running through the initial steps involved in creating a process definition. You can rerun the tour at any time by clicking the icon in the toolbar.

When you open the BPMN editor to create a new process definition, the canvas already contains a Start Event. Clicking on any event on the canvas frames the event icon with a dotted line and reveals a number of controls.

The controls below the icon allow you to delete the BPMN object, or change in to another object in the same group. For example, you can change a Start event to a Start timer event. The controls to the right of the icon allow you to specify the next object type in the process. The list presented includes only those object types that are valid in the sequence after the current object. There are also controls that allow you to create flows connecting other existing events in your diagrams, and to annotate the event.

There are two ways of adding BPMN objects to your process:

- Use the controls that appear when you click on a current object icon. Using this method will create a valid connector between the current event icon and the new event icon.
- Drag and drop an object icon from the palette. In this case you add flows to the current event icons in the process yourself by picking the icons from the palette.

The following object groups are shown in a collapsible list in the palette. The groups consist of all the objects available in the BPMN 2.0 specification, and additional Alfresco Activiti extensions such as the Publish to Alfresco task, Publish to Box, Publish to Google Drive.

2.7.1. Start events

A start event indicates where a process starts. You can define a start event in one of the following ways:

- Start on the arrival of a message
- Start at specific time intervals
- Start as a result of an error
- Start when a specific signal is raised
- Start on no specific trigger

In the XML representation, the type start event is specified as a sub-element.

Start events are always catching: a start event waits until a specific trigger occurs.

None start event

A start event with an unspecified trigger. BPMN 2.0 refers to this as a none start event. It is visualized as a circle with no icon.

A none start event can have a *start form*. If so, the start form will be displayed when selecting the process definition from the *processes* list. Note that a process instance is not started until the start form is submitted. A none start event without a form will simply have a button displayed to start the process instance.

Note	A subprocess always has a none start event.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this element instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Process Initiator	The process variable in which the user ID of the initiator of this instance should be stored.
Form key	A key that provides a reference to a form. This property is available for compatibility with Activiti community, but should not be used directly when using the Alfresco Activiti Forms. Use the *Referenced form* property instead.
Referenced form	A form reference.

Property	Description
Form properties	A form definition with a list of form properties. Form properties are the way forms are defined in the community version of Activiti. Configuring them has no impact on the rendered form in the Alfresco Activiti, the *Referenced form* property should be used instead.

Start timer event

A timer start event initiates a process instance at specific time. You can use it both for processes which must start only once and for processes that must start in repeated time intervals.

It is visualized as a circle with a clock icon.

Note that a process instance started by a timer start event can't have a start form, as it is started by the system. Similarly, it does not have a process initiator like a *none start event*. As such when assigning tasks later on in the process definition, keep in mind that the assignment *'assigned to process initiator'* will not work.

Note	A subprocess can't have a timer start event.

Property	Description
Id	A unique identifier for this instance.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Time Cycle	A timer cycle defined in http://en.wikipedia.org/wiki/ISO_8601 format, for example: `R3/PT10H`.

Property	Description
Time Date in ISO-8601	A point in time defined as a http://en.wikipedia.org/wiki/ISO_8601 date, for example: `2015-04-12T20:20:32Z`.
Time Duration	A period of time defined as a http://en.wikipedia.org/wiki/ISO_8601 duration, for example: `PT5M`.

Start signal event

A signal start event starts a process instance using a named signal. The signal is fired from a process instance using the intermediary signal throw event (or programmatically through the java or REST API). In both cases, a process instance for any process definitions that have a signal start event with the same name are started. You can select a synchronous or asynchronous start of the process instances.

A signal start event is visualized as a circle with a triangle inside. The triangle is white inside.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Signal reference	The name of the signal that initiates this event. Note that signal references are configured on the root level of the process instance and then linked to the signal start event via this property. To configure it, deselect any other element and click the *Signal definitions* property.

Start message event

A message start event starts a process instance using a named message. It is mainly used for starting process instances from external systems.

It is depicted as a circle with an envelope icon inside. The envelope is white inside.

When you deploy a process definition with one or more message start events, consider the following points:

- The name of the message start event must be unique across the whole process definition. Alfresco Activiti will throw an exception on deployment of a process definition with two or more message start events that reference the same message or with two or more message start events that reference messages with the same name.
- The name of the message start event must be unique across all deployed process definitions. Alfresco Activiti will throw an exception on deployment of a process definition with one or more message start events that reference a message with the same name as a message start event already deployed in a different process definition.
- When a new version of a process definition is deployed, the message subscriptions of the previous version are canceled. This is also true for message events that are not present in the new version.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.

Property	Description
Message reference	The name of the message that initiates this event. Note that messages are configured on the root level of the process instance and then linked to the message start event via this property. To configure it, deselect any other element and click the *'Message definitions'* property.

Start error event

An error start event triggers an event Sub-Process. An error start event can't be used for starting a process instance.

It is visualized as a circle with lightning icon inside. The icon is white inside.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Error reference	The name of the error that initiates this event. This reference needs to match the error identifier thrown by the event that throws the particular error.

2.7.2. Activities

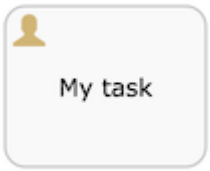
An activity describes a single item of work to be performed in a process. Alfresco Activity provides some Activity types that are additional to those described in the BPMN 2.0 specification.

An activity is always visualized as a rectangle with rounded corners.

User task

A user task enables you to model work to be done by a human actor. When process execution arrives at a user task in the process definition, it creates a new task in the task list of the assignee or assignees defined in the task.

A user task is depicted as a rounded rectangle with a user icon on the top-left corner.



Property	Description
Id	A unique identifier for this element
Name	A name for this element.
Documentation	A description of this element.
Assignment	Configures to who this task should be assigned. It is possible to use *Fixed Values* (advanced usage: these are Activiti expressions, for example by invoking a class or Spring bean) or use the *Identity Store* option. It is recommended to use *Identity Store* to select groups and users in the system: ***Assigned to process initiator*** The user that started the process instance will be the assignee of this task. ***Assigned to process initiator's (primary) group manager*** The group manager of the user that started the process instance will be the assignee of this task. ***Assigned to single user*** A single user who will be the assignee of the task. This user will see the task in their *Involved tasks* task list. It is possible to reference a user that was selected in a previous form field (tab *Form field*). ***Assigned to group manager***

Property	Description
	The group manager of the user will be the assignee of the task. Only users that have a primary group defined will have a group manager. To define a primary group, go to **Identity Management** > **Users** > **Select an action** > **Change primary group**. ***Candidate users*** One or more users as the *candidate(s)* of the group. The task will show up in their *Queued tasks* task list. The task is not yet assigned to them. They first have to *claim* the task, which will make that one user the assignee. The other users won't see that task in a task list anymore. It is possible to reference users that were selected in a previous form field (tab *Form field*). ***Candidate groups*** One or more groups whose members will be the *candidate* of the group. The task will show up in their *Queued tasks* task list. The task is not yet assigned to them. They first have to *claim* the task, which will make that one user the assignee. The other users won't see that task in a task list anymore. It is possible to reference groups that were selected in a previous form field (tab *Form field*). ***Allow process initiator to complete task*** When checked, the user that started the process instance (process initiator) can complete the task. This is checked by default.
Referenced form	Allows to configure or create the form for this task. This form (also called _task form) will be rendered when the task is shown in the task list of the user. A user task typically always has a form defined.
Form key	This is a property that exists for compatibility with Activiti community. When using Alfresco Activiti to work with task lists and forms, do not set this property.
Form properties	This is a property that exists for compatibility with Activiti community. When using Alfresco Activiti to work with task lists and forms, do not set this property.

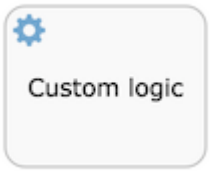
Property	Description
Due date	Allows to configure a due date for the task. In the task list, tasks can be sorted by due date to see which tasks are needed to be completed the soonest. The possible ways of configuring are: ***No due date*** This is the default value. ***Expression definition (Advanced)*** Uses an Activiti expression to resolve the due date (for example, this expression could call a Spring bean). ***Fixed duration after task creation*** Allows to configure an amount of time, starting from the creation of the task. ***Based on field*** Allows to configure the due date based on a previous field in the process instance, by adding or subtracting a certain amount of time. ***Based on variable*** Allows to configure the due date based on a variable previously declared in the process instance, by adding or subtracting a certain amount of time.
Allow email notifications	When enabled, an email will be sent to the assignee when the task is created.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be created as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.

Property	Description
Execution listeners	Execution listeners configured for this instance. An execution listener lets you execute Java code or evaluate an expression when an event occurs during process execution.
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	(Used with Multi-Instance type) The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.

Service task

Use a service task to invoke an external Java class or execute an expression (for example to call a Spring bean).

A service task is visualized as a rounded rectangle with a cog icon inside.



Property	Description
Id	A unique identifier for this element instance.
Name	A name for this element.
Documentation	A description of this element.
Class	The name of the Java class that implements your service task. Your class must implement `JavaDelegate` or `ActivityBehavior`. For more information on methods of invoking Java logic from a service task see the Activiti Developer Guide
Expression	An expression that either executes logic in the expression itself (for example *${execution.setVariable('myVar', 'someValue')}*) or calls a method on a bean known by the Activiti engine (for example *${someBean.callMethod}*). You can pass parameters (like the current *execution*) to the method in the expression. For more information on methods of invoking Java logic from a service task see the Activiti Developer Guide.
Delegate expression	An expression that resolves to an object which is of a class that implements `JavaDelegate` or `ActivityBehavior`. For example *${someBean}* resolves to a bean that implements said interfaces.
Class	Field extensions for the service task.

Property	Description
Result variable name	The name of a process variable in your process definition in which to store the result of this service task. This is only valid when using an *expression*.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Multi-Instance type	Determines if this task is performed multiple times and how. For more information on multi-instance, see the Activiti Developer Guide. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.

Property	Description
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.

For a service task it is recommended to make them asynchronous. For example, suppose a service task is called after the user completes a form. When the service task is synchronous, the logic will be executed during the completion action of the user. This means the user has to wait until this logic is finished to have the UI refreshed. Often, this is not needed or wanted. By making the service task asynchronous, the UI will be refreshed when the task is completed. The logic will be executed later.

Script task

A script task defines a JavaScript script or other script language (JSR-223 compatible language) that is executed when a process instance executes this step.

A script task is visualized as a rounded rectangle with a paper icon inside.

Property	Description
Script format	The JSR-223 name of the scripting engine your script is written for. By default, Alfresco Activiti supports *javascript* and *groovy* formats.

Property	Description
Script	The actual script that will be executed.
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Variables	In the script, it is possible to set new process variables (using *'execution.setVariable('myVariable', 'myValue')'*), however these won't show up automatically in dropdowns later on (like the sequence flow condition builder, forms, etc.). To make them show up, configure this property with the variables that are set or 'exported' by this script task.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.

Property	Description
Multi-Instance type	Determines if this task is performed multiple times and how. For more information on multi-instance, The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.

Business rule task

A Business rule task executes one or more rules. It is mainly there for compatibility with Activiti community. In Alfresco Activiti, Alfresco recommends to use the Decision tables. See Business Rules - Decision Tables for more information on that.

A business rule is depicted as a rounded rectangle with a table icon in the top-left corner.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Rules	A comma-separated list of rules to include or exclude in this task.
Input variables	A comma-separated list of process variables to be used as input variables to your rules.
Exclude	If you check Exclude only rules that you have not specified in Rules will be executed. If the Exclude is unchecked, only the rules you have specified in Rules will be executed.
Result variable	The name of a process variable in your process definition in which to store the result of this task. the result variable is returned as a list of objects. If you do not specify a result variable name, the default name `org.activiti.engine.rules.OUTPUT` is used.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.

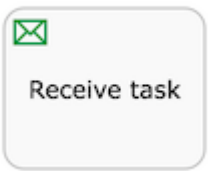
Property	Description
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Multi-Instance type	Determines if this task is performed multiple times and how. For more information on multi-instance, see the Activiti Developer Guide. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.

Property	Description
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.

Receive task

A Receive Task waits for the arrival of an external trigger. This trigger is sent programmatically (via Java or REST API). For process to process triggering, use the signal events.

A receive task is visualized as a rounded rectangle with an envelope icon in the top-left corner.



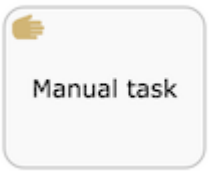
Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Variables	When the API is used to trigger the continuation of the process instance, a set of variables can be passed. However, these won't appear automatically in drop-down lists later (like the sequence flow condition builder, forms, etc.). To make them appear, this property needs to be configured with those variables that are set or exported by the script task.

Property	Description
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced)Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Multi-Instance type	Determines if this task is performed multiple times and how. For more information on multi-instance, see the *Activiti Developer Guide*. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.

Property	Description
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.

Manual task

A Manual Task defines a task that is external to Activiti. You use it to model work done which the Activiti engine does not know of. A manual task is handled as a pass-through activity, the Activiti engine automatically continues the process from the instant process execution arrives at a manual task activity.



Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.

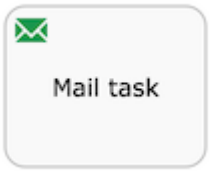
Property	Description
Exclusive	(Advanced)Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.

Property	Description
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the *Activiti Developer Guide*.

Mail task

You can enhance your business process with this automatic mail service task that sends emails to one or more recipients. The task supports normal email features such as cc lists, bcc lists, and HTML content.

The mail task is depicted as a rounded rectangle with an envelope icon in the top-left corner.



Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
To	The recipient of the e-mail. You can specify multiple recipients in a comma-separated list. When using a fixed value, this can be an expression. It is also possible, like with the user task, to use the *Identity store* option here to pick users that are known in the system or to reference people that were selected in form fields prior to this email task.
From	The sender's email address. If you do not specify this, the default configured system-wide setting *from* address is used. This can be an expression.
Subject	The subject of this email. This can be an expression.

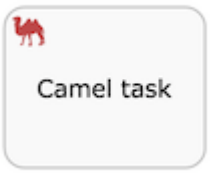
Property	Description
Cc	The cc list for this email. You can specify multiple recipients in a comma-separated list. This can be an expression.
Bcc	The bcc list for this email. You can specify multiple recipients in a comma-separated list. This can be an expression.
Text	The text content of this email. You can specify this as well as HTML to support email clients that do not support rich content. The client will fall back to this text-only alternative.
Html	The HTML content of this email.
Charset	The charset for this email. By default UTF8 will be used.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.

Property	Description
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the *Activiti Developer Guide*.

Camel task

You use the Camel task to send messages to, and receive messages from Apache Camel.

A camel task is visualized as a rounded rectangle with a camel icon in the top-left corner.



You can find more information on Apache Camel here. Note that Camel is by default not installed and would need to be added by the system admin.

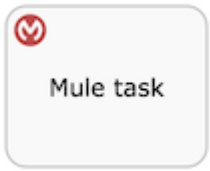
Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Camel context	A camel context definition. If you do not specify a context, the default Camel context is used.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.

Property	Description
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.

Mule task

Use the Mule task to send messages to the Mule ESB (Enterprise Service Bus).

A mule task is visualized as a rounded rectangle with the Mule logo in the top-left corner.



You can find more information on Mule ESB here. Note that Mule is by default not installed and would need to be added by the system admin.

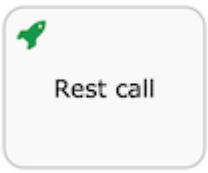
Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Endpoint url	The Mule endpoint you want to send your message to.
Language	The language you want to use to evaluate the payloadExpression, for example juel.
Payload expression	An expression for the message's payload.
Result variable	The name of the variable to store the result of the invocation.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.

Property	Description
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.

Rest call task

The rest call task is used to communicate with a REST endpoint. The endpoint can be defined in the process definition, or it can be defined company-wide by an administrator. In the latter case, a logical name is all that is needed.

A rest call task is visualized as a rounded rectangle with a rocket icon the top-left corner.



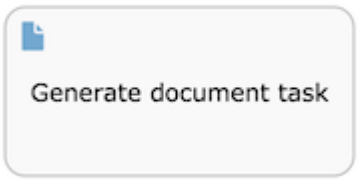
Note that the REST call task always is executed asynchronously.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Endpoint	Defines which REST endpoint to call. It is an endpoint defined company-wide by the administrator (simply select a logical name in the dropdown) or a URL. You can also use previously defined form fields or variables to build up the URL.
Request mapping	Allows to construct the actual request. HTTP GET represents the URL parameters whereas POST/PUT is the json body that is created when the request is sent. You can also use fixed values, form fields, or variables defined prior to this activity.
Response mapping	Maps the JSON response from the REST endpoint to process variables. You can use a nested notation (e.g. *prop1.prop2.prop3*) for mapping values. The mapped response values can be used as variables in further steps of the process.

Generate document task

The generate document task generates a document (Word or PDF) and stores the reference to the document as a process variable. The document is based on a (Word) template that describes how the document needs to be rendered, using process variables and various constructs (such as if-clauses and loops). See the *Activiti Developer's Guide* on how to use the generate document task.

A generate document task appears as a rounded rectangle with a document icon on the top-left corner.



Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Template	The template upon which is used to generate the document. It can be uploaded as part of the process definition, or can be defined company-wide by an administrator and reused by various process definitions.
Output format	The document format, PDF or Word.
Document variable	This is the process variable in which the reference to the generated document is stored.

Decision task

You use a decision task to select a decision table while designing your process model. A decision table enables you to define a set of business rules that will be applied when it's executed. See the Business Rules - Decision Tables section for more information.

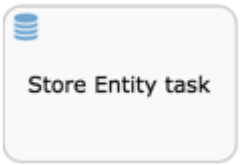
A decision task is depicted as a rounded rectangle with a table icon the top-left corner.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Reference decision table	Defines the actual decision table that will be executed. The decision table can be part of the process definition (a so-called 'embedded' decision table) or defined on itself (a so-called 'reusable' decision table).
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.

Property	Description
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.
Is for compensation	If this activity is used for compensating the effects of another activity, you can declare it to be a compensation handler. For more information on compensation handlers see the Activiti Developer Guide.

Store Entity task

Use the Store entity task to update data models or entities with process values such as variables or form fields. The updated entities can then be mapped to variables and used while creating processes.



Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Attribute mapping	Attributes mapped for this element instance. Click to invoke the Change value for "Attribute Mapping" dialog, where you can map entities or Data Models with form fields and variables used in your process. See the Data Models section for more details.

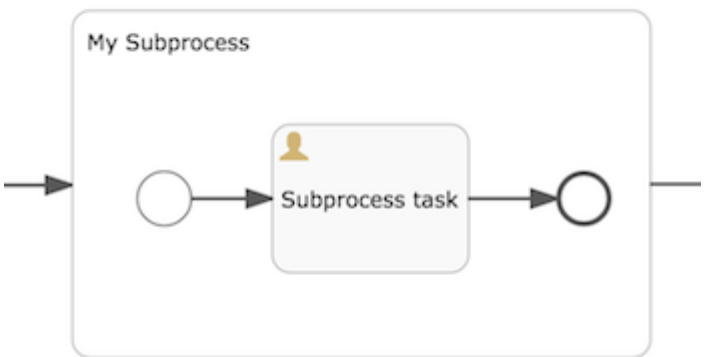
2.7.3. Structural components

You use structural components to group multiple components in a sub process to reuse in a parent process definition, and to embed and call other process definitions from inside your own process.

Sub-process

A sub process is a single activity that contains activities, gateways, and events which form a process. A sub process is completely embedded inside a parent process.

A sub-process is visualized as a rounded rectangle:



You can use a sub process to create a new scope for events. Events that are thrown during execution of the sub process, can be caught by Boundary events on the boundary of the sub process, creating a scope for that event limited to just the sub process.

Sub-processes must have the following characteristics:

- A sub process has exactly one none start event. No other start event types are permitted. A sub process must have at least one end event.
- Sequence flow cannot cross sub process boundaries.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.

Property	Description
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.

Collapsed sub-process

You use a collapsed sub-process to add an existing process from your available process definitions as a sub-process to the process definition you are currently editing.

When you drag a collapsed sub-process from the palette to your canvas, and click on the Referenced Subprocess property, you are presented with a visual list of the process definitions you have access to. You can choose from the list, and the chosen process will be added to the current process definition. Note the the process chosen must have exactly one none start event, and no other start event type, and it must have at least one end event.

Not that during process instance execution, there is no difference between a collapsed or embedded sub-process. They both share the full process instance context (unlike the *call activity*).

Note that when you click on the plus icon in a collapsed sub-process, the BPMN editor will open the referenced sub-process definition.

A collapsed sub-process is visualized as a rounded rectangle with a plus icon inside.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element instance.
Referenced Subprocess	The process definition this collapsed sub-process contains.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.

Property	Description
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.

Event sub-process

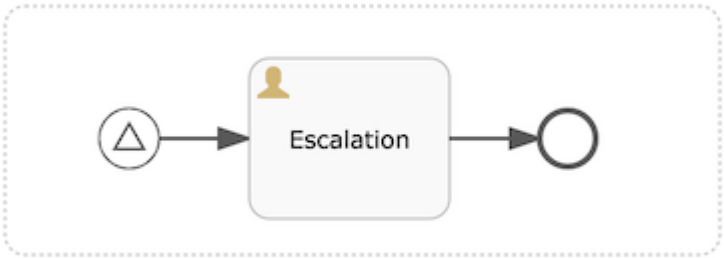
An event sub-process is a sub-process that is triggered by an event. You can use an event sub-process in your main process, or in any sub-process.

The event sub-process start event defines the event to be handled by the sub-process, so the type of start event you use must have an event associated with it – none start events are not supported but the event sub-processes. Your event sub-process can be started by a start message event, start signal event or a start error event. The subscription to the start event is created when the scope, process instance or sub-process, hosting the event sub-process is created. The subscription is removed when the scope is destroyed.

Your event sub-process does not have any incoming or outgoing sequence flows. An event sub-process is triggered by an event, so there can be no incoming sequence flow.

The best way to look at an event subprocess is as a 'method' or 'routine' that is called when something happens, and handle it appropriately.

An event sub-process is visualized like a sub-process with a dashed border.



Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.
Exclusive	(Advanced)Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.

Property	Description
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.

Call activity

A call activity is used to execute another process definition as part of the current process instance.

The main difference between a sub-process and a call activity is that the call activity does not share context with the process instance. Process variables are explicitly mapped between the process instance and the call activity.

A call activity is visualized as a rounded rectangle with a thick border.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Called element	This is the identifier of the process definition that should be called.
In parameters	Configures the process variables that are mapped into the called process instance when it’s executed. It’s possible to copy values directly (using the 'source' attribute) or with an expression (using the 'source expression' attribute) in a 'target' variable of the called process instance.
Out parameters	Configures the process variables that are mapped from the called process instance into the parent process instance.

Property	Description
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Execution listeners	Execution listeners configured for this instance. An execution listeners is a piece of logic that is not shown in the diagram and can be used for technical purposes.
Multi-Instance type	Determines if this task is performed multiple times and how. The possible values are: ***None*** The task is performed once only. ***Parallel*** The task is performed multiple times, with each instance potentially occurring at the same time as the others. ***Sequential*** The task is performed multiple times, one instance following on from the previous one.
Cardinality (Multi-instance)	The number of times the task is to be performed.
Collection (Multi-instance)	The name of a process variable which is a collection. For each item in the collection, an instance of this task will be created.
Element variable (Multi-instance)	A process variable name which will contain the current value of the collection in each task instance.

Property	Description
Completion condition (Multi-instance)	A multi-instance activity normally ends when all instances end. You can specify an expression here to be evaluated each time an instance ends. If the expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends.

2.7.4. Gateways

You use gateways to control the flow of execution in your process.

In order to explain how Sequence Flows are used within a Process, BPMN 2.0 uses the concept of a token. Tokens traverse sequence flows and pass through the elements in the process. The token is a theoretical concept used to explain the behavior of Process elements by describing how they interact with a token as it “traverses” the structure of the Process. Gateways are used to control how how tokens flow through sequence flows as they converge and diverge in a Process.

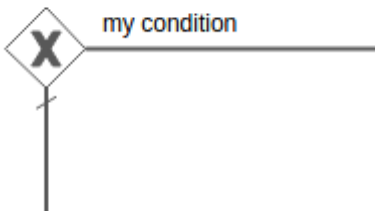
As the term gateway suggests, it is a gating mechanism that either allows or prevents passage of a token through the Gateway. As tokens arrive at a Gateway, they can be merged together on input and/or split apart on output from the Gateway.

A gateway is displayed as a diamond, with an icon inside. The icon depicts the type of gateway.

Exclusive gateway

You use an exclusive gateway to model a decision in your process. When execution arrives at an exclusive gateway, the outgoing sequence flows are evaluated in the order in which they are defined. The first sequence flow whose condition evaluates to true, or which does not have a condition set, is selected and the process continues.

An exclusive gateway is visualized as a diamond shape with an X inside.



Note that if no sequence flow is selected, an exception will be thrown.

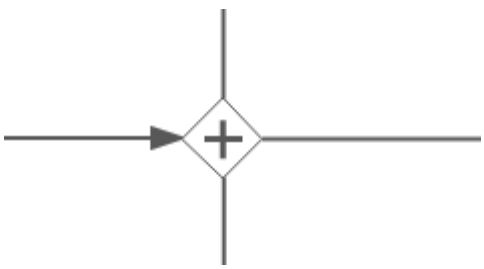
Property	Description

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it's not important to have the task immediately ready.
Exclusive	(Advanced) Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Flow order	Select the order in which the sequence flow conditions are evaluated. The first sequence flow that has a condition that evaluates to true (or has no condition) will be selected to continue.

Parallel gateway

You use a parallel gateway to model concurrency in a process. It allows you to fork multiple outgoing paths of execution or join multiple incoming paths of execution.

A parallel gateway is visualized as a diamond shape with a plus icon:



In a fork, all outgoing sequence flows are followed in parallel, which creates one concurrent execution for each sequence flow.

In a join, all concurrent executions arriving at the parallel gateway wait at the gateway until an execution has arrived for every incoming sequence flow. Then the process continues past the joining gateway. Note that the gateway simply wait until the required number of executions has been reached and does not check if the executions are coming from different incoming sequence flow.

A single parallel gateway can both fork and join, if there are multiple incoming and outgoing sequence flow. The gateway will first join all incoming sequence flows, before splitting into multiple concurrent paths of executions.

Unlike other gateways, the parallel gateway does not evaluate conditions. Any conditions defined on the sequence flow connected with the parallel gateway are ignored.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.
Exclusive	(Advanced)Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Flow order	Select the order in which the sequence flow conditions are evaluated. Note that for a parallel gateway this is not important, as conditions are not evaluated.

Inclusive gateway

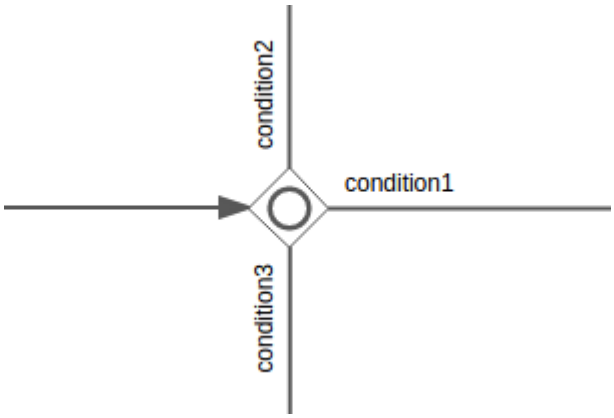
You use an inclusive to join and fork multiple sequence flows based on conditions.

Like an exclusive gateway you can define conditions on outgoing sequence flows and the inclusive gateway will evaluate them, but an inclusive gateway can take more than one sequence flow, like the parallel gateway.

All outgoing sequence flow conditions are evaluated. Every sequence flow with a condition that evaluates to true, is followed in parallel, creating one concurrent execution for each sequence flow.

The join behavior for an inclusive gateway is more complex than the parallel gateway counterparts. All concurrent executions arriving at the inclusive gateway wait at the gateway until executions that *can* reach the inclusive gateway have reached the inclusive gateway. To determine this, all current executions of the process instance are evaluated, checking if there is a path from that point in the process instance to the inclusive gateway. (ignoring any conditions on the sequence flow). When one such execution is found, the inclusive gateway join behavior does not activate.

An inclusive gateway is visualized as a diamond shape with a circle icon inside:



Note that an inclusive gateway can have both fork and join behavior, in which case there are multiple incoming and outgoing sequence flows for the same inclusive gateway. The gateway will join all incoming sequence flows that have a process token, before splitting into multiple concurrent paths of executions for the outgoing sequence flows that have a condition that evaluates to true.

Property	Description
Id	A unique identifier for this element instance.
Name	A name for this element instance.
Documentation	A description of this element instance.
Asynchronous	(Advanced) Define this task as asynchronous. That is, the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.

Property	Description
Exclusive	(Advanced) Define this task as exclusive. That is, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Flow order	Select the order in which the sequence flow conditions are evaluated. This is of less importance as for the exclusive gateway, as all outgoing sequenceflow conditions will be evaluated anyway.

Event based gateway

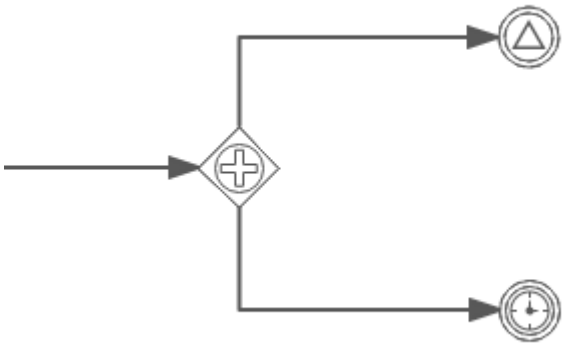
You use an event gateway to route process flow based on events.

Each outgoing sequence flow of the event gateway must be connected to an intermediate catching event. When process execution reaches an event gateway execution is suspended, and for each outgoing sequence flow, an event subscription is created. The flow for the event that occurs first, will be followed.

Outgoing sequence flows connect to an event gateway are never "executed", but they do allow the process engine to determine which events an execution arriving at an Event-based Gateway needs to subscribe to. The following restrictions apply to event gateways:

- The gateway must have two or more outgoing sequence flows.
- An Event-based Gateway can only be followed by intermediate catching events. Receive tasks after an event gateway are not supported by Activiti.
- An intermediate catching event connected to an event gateway must have a single incoming sequence flow.

An event gateway is visualized as a diamond shape with a plus icon inside. Unlike the parallel gateway, the plus icon is not colored black inside:



Property	Description
Id	A unique identifier for this element instance.
Name	A name for this element instance.
Documentation	A description of this element instance.
Asynchronous	(Advanced) Define this task as asynchronous. This means the task will not be executed as part of the current action of the user, but later. This can be useful if it’s not important to have the task immediately ready.
Exclusive	(Advanced)Define this task as exclusive. This means that, when there are multiple asynchronous elements of the same process instance, none will be executed at the same time. This is useful to solve race conditions.
Flow order	Select the order in which the sequence flow conditions are evaluated.

2.7.5. Boundary events

You use boundary events to handle an event associated with an activity. A boundary event is always attached to an activity.

While the activity the boundary event is attached to 'is active' (meaning the process instance execution is currently executing it right there), the boundary event is listening for a certain type of trigger. When the event is caught, the activity is either interrupted and the sequence flow going out of the event is followed (interrupting behavior) or a new execution is created from the boundary event (non-interrupting behavior).

Boundary timer event

A boundary timer event puts a timer on the activity it is defined on. When the timer fires, the sequence flow going out the boundary event is followed.

A boundary timer event is visualized as a circle with a clock icon inside:

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Cancel activity	Defines if the boundary event interrupts the activity is defined upon or not.
Time Cycle	A timer cycle defined in http://en.wikipedia.org/wiki/ISO_8601 format, for example: `R3/PT10H`.
Time Date in ISO-8601	A point in time defined as a http://en.wikipedia.org/wiki/ISO_8601 date, for example: `2015-04-12T20:20:32Z`.

Property	Description
Time Duration	A period of time defined as a http://en.wikipedia.org/wiki/ISO_8601 duration, for example: `PT5M`.

Boundary error event

A boundary error event catches an error that is thrown within the boundaries of the activity the event is based on and continues process execution from the event.

A boundary error event is always interrupting.

A boundary timer event is visualized as a circle with a lightning icon inside:

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Error reference	The identifier of the error to catch.

Boundary signal event

A boundary signal event listens to a signal being fired (from within the process instance or system-wide) while the activity upon which the event is defined is active.

A boundary signal event is visualized as a circle with a triangle icon inside:

Property	Description

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Signal reference	The signal to listen to. Signals are defined on the root process definition level and are linked with this property.

Boundary message event

A boundary message event listens to a message being received while the activity upon which the event is defined is active.

A boundary message event is visualized as a circle with an envelope icon inside:

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Message reference	The message to listen to. Messages are defined on the root process definition level and are linked with this property.

Boundary cancel and compensation event

The boundary cancel and compensation event are currently experimental features. See http://activiti.org/userguide/index.html#bpmnBoundaryCancelEvent for more information on them.

2.7.6. Intermediate catching events

An intermediate catching event is a step in the process where the process needs to wait for a specific trigger (in BPMN this is described as 'catching' semantics).

An intermediate event is displayed as two concentric circles containing an icon. The icon shows the type of intermediate event:

Conceptually, the intermediate catch events are close to the boundary events, with that exception they don't define a scope (the activity) for when the event is active. An intermediate catch event is active as long as the trigger hasn't happened. A boundary event on the other hand can be destroyed if the activity completed.

All the supported intermediate catch events are configured similar to their boundary event counterparts.

2.7.7. Intermediate throwing events

An intermediate throw event is used to explicitly throw an event of a certain type.

Currently, two types are supported:

- The **none intermediate throwing event**. No event is thrown. This is mainly used as a marker in the process definition (for example to attach execution listeners that are used to indicate somehow that some state in the process has been reached).
- The **signal intermediate throwing event**. Throws a signal event that will be caught by boundary signal events or intermediate signal catch events listening to that particular signal event.

An intermediate event is displayed as two concentric circles which may contain an icon. If present, the icon shows the type of intermediate event. A throwing none event contains no icon.

2.7.8. End events

You use an end event to signify the end of a process or sub-process, or the end of a path in a process or sub-process.

In a subprocess or process instance, only when all executions have reached an end event will the subprocess be continued or the whole process instance ended.

An end event is displayed as thick black circle which may contain an icon. If present, the icon shows the type of end event. A none end event has no icon.

None end event

A none end event ends the current path of execution.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this event.

Error end event

You use the end error event to throw an error and end the current path of execution.

The error can be caught by an intermediate boundary error event that matches the error. If no matching boundary error event is found, an exception will be thrown

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this instance.

Property	Description
Error reference	The error identifier. This is used to find a matching catching boundary error event. If the name does not match any defined error, then the error is used as the error code in the thrown exception.

Terminate end event

When a terminate end event is reached, the current process instance or sub-process will be terminated. Conceptually, when an execution arrives in a terminate end event, the first scope (process or sub-process) will be determined and ended. Note that in BPMN 2.0, a sub-process can be an embedded sub-process, call activity, event sub-process or transaction sub-process. This rule applies in general, for example, when there is a multi-instance call activity or embedded subprocess, only that instance will be ended, the other instances and the process instance are not affected.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this.

Cancel end event

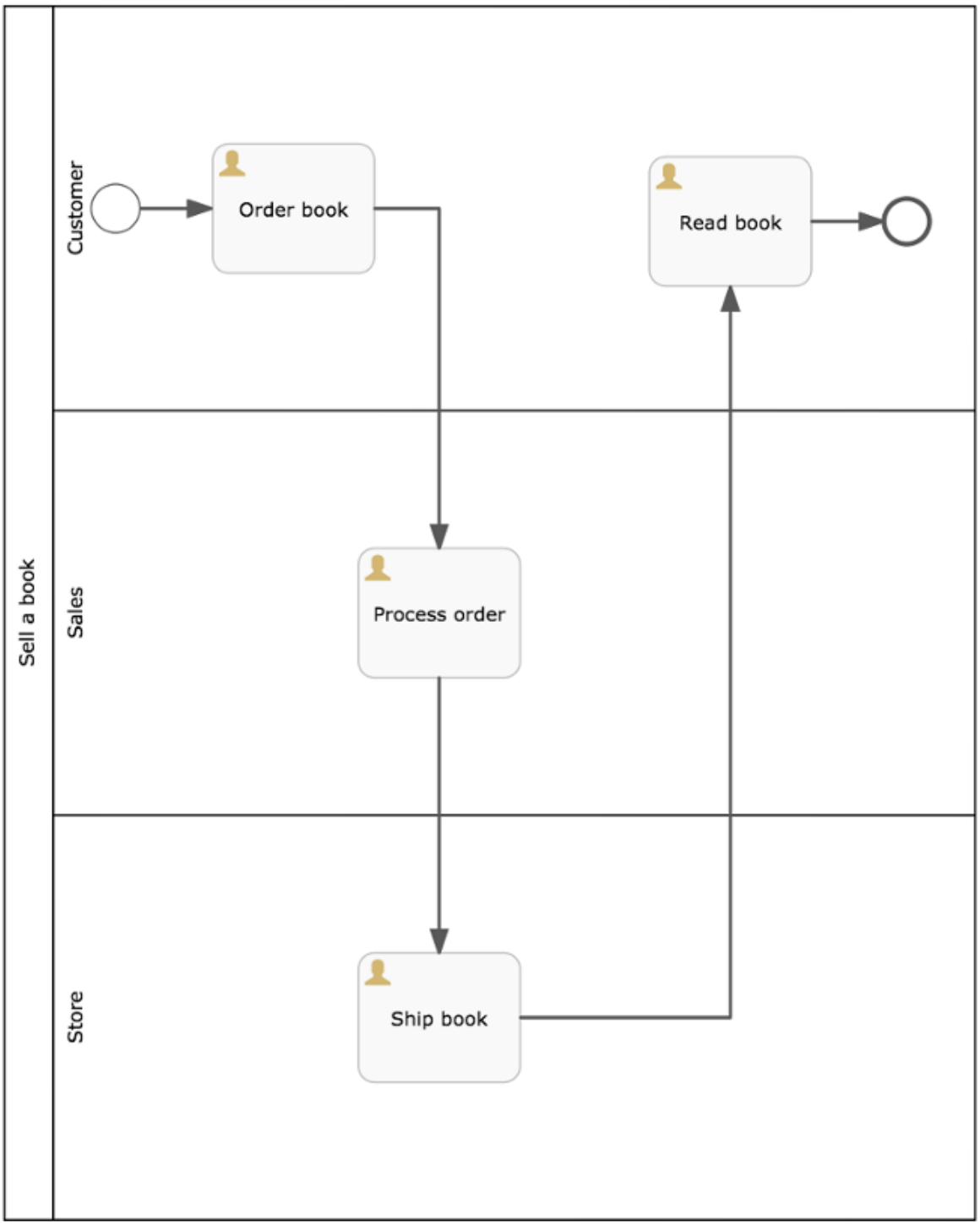
The cancel end event ends the current path of execution and throws a cancel event that can be caught on the boundary of a transaction subprocess.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Execution listeners	Execution listeners configured for this.

2.7.9. Swimlanes

You use swimlanes to display activities in your process divided by business function or participant group. A process definition can have one swimlane diagram containing one pool, which in turn contains one or more lanes. The pool represents the whole process, and each lane corresponds to a business function or participant group.

For example, the process of selling a book consists of several activities: ordering a book, processing the order, shipping the book, and reading the book. However, the activities are performed by participants in different groups: by the customer, by the sales department, by the warehouse, or store. In the following diagram, process definitions have one pool called Sell a book with three lanes: Customer, Sales, and Store. The process sequence flow moves between lanes in the pool as the order progresses.



When you drag a pool to your process diagram, it creates an unnamed pool containing one unnamed lane. You can add lanes by dragging a lane icon from the palette to the canvas. When you hover over the name box of the pool, the whole pool border turns green, indicating the lane will be added to the pool when you release the mouse button.

2.7.10. Artifacts

You use artifacts to provide additional information about the process. The BPMN editor supports the text annotation artifact which associates additional text to an element in your process, or to the process itself. The text does not influence the execution of a process and is is provided by the process designer to give information to the user of the process.

Text annotation

You can set the following properties in the property sheet:

Property	Description
Id	A unique identifier for this element instance
Name	A name for this element instance
Documentation	A description of this element instance
Text	The text you want to display in your annotation

2.7.11. Alfresco

Use this section for actions specific to Alfresco content store:

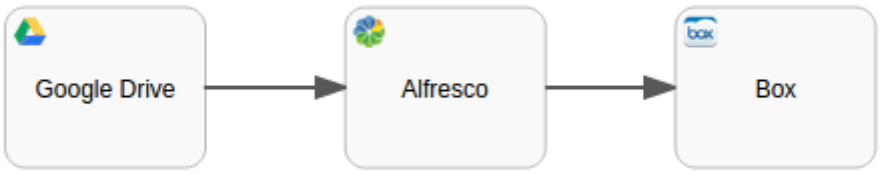
- Publish to Alfresco task
- Rerieve Alfresco Properties
- Update Alfresco Properties
- Call Alfresco Action

Publish to Alfresco task / Box / Google Drive

The publish task enables you to publish items that were created or modified during process instance execution to a content store. Currently, the following content stores are supported:

- Alfresco One version 4.0 or later, or Alfresco in the Cloud
- Box
- Google Drive

A publish task is depicted as a rounded rectangle with the icon of the content store on the top-left corner.



Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Alfresco / Box / Google Drive Content	Configures what content to publish. You can select a previously defined form field or all the content that was updated during the process instance execution.
Alfresco / Box / Google Drive Destination	Configures where the content will be published to. You can publish the content using the process initiator or a specific user (this is important when it comes to permissions in the content store).

Retrieve Alfresco Properties

The Retrieve Alfresco Properties option enables you to retrieve content-specific properties from Alfresco One and map it to a form field or variable, for example properties of a document linked from Alfresco. You can retrieve document information after a document is added or referenced via the Attachment form field in Alfresco Connector.

Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.

Property	Description
Alfresco properties	Retrieves Alfresco properties for content stored in the form editor or variable, and allows mapping them.

Update Alfresco Properties

The Update Alfresco Properties option enables you to update content-specific properties in Alfresco via a form field or variable. For example, you can update properties of a document linked from Alfresco via a form attachment field, or process variable.

The Properties sheet displays the same fields as Retrieve Alfresco properties, except that is is used for updating properties rather than retrieving.

Call Alfresco Action

The Call Alfresco Action enables you to invoke the standard Alfresco One actions from Activiti.

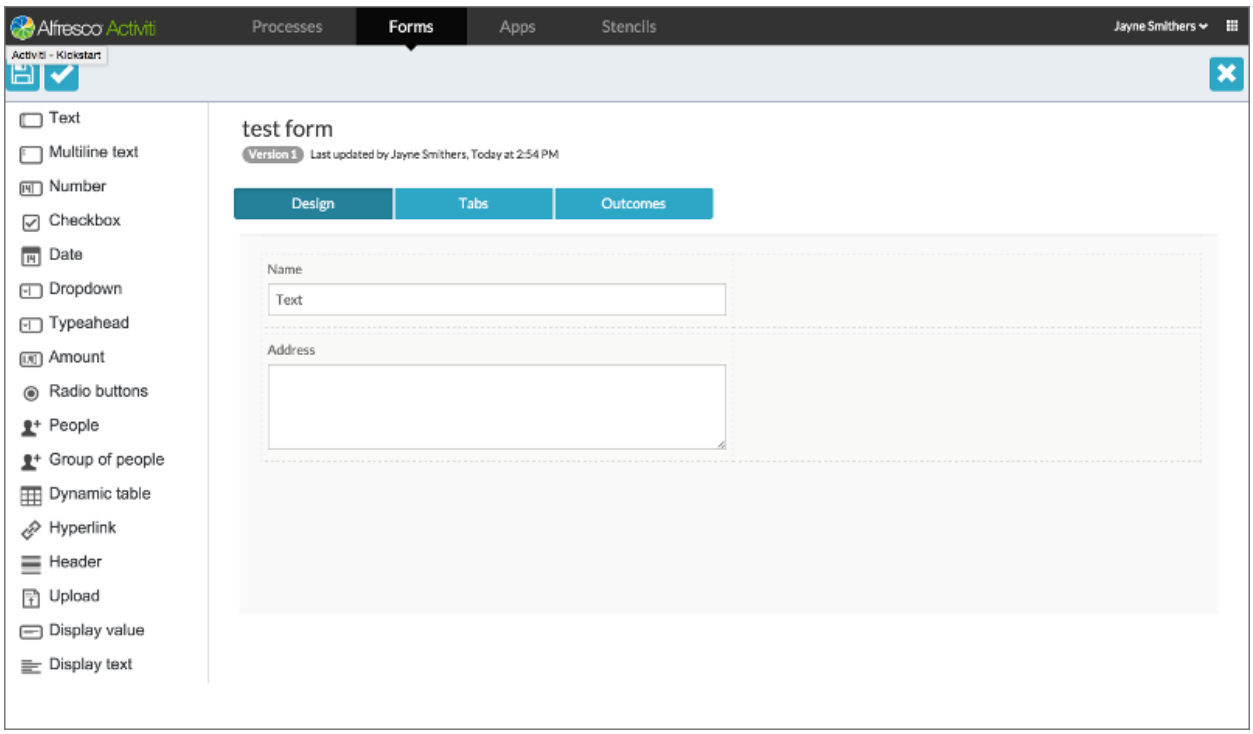
Property	Description
Id	A unique identifier for this element.
Name	A name for this element.
Documentation	A description of this element.
Content	Retrieves Alfresco properties for content stored in the form editor or variable.
Act as	Identity of the caller: Process initiator or Specific User. Selecting **Specific User** lets you select a different user.
Repository	Changes the repository account. For example: Alfresco One One, Alfresco in the Cloud.
Action	Lists a range of actions specific to Alfresco One. Select the options to make changes to the default name and value depending on your requirement. The options are as follows: ***extract-metadata***

Property	Description
	Extracts embedded metadata from files and added to the file properties. Alfresco one supports Microsoft Office document properties, LibreOffice, and a number of other formats. *move* Moves the files and subfolders to the locations of your choices in Share if you edit the following value with the exact location of your document in Share: *workspace://SpacesStore/<ID>* *add aspect* Adds a property aspect to files for additional behaviours or properties. *specialise-type* Changes a file's content type, if applicable. For example, you can changes a standard file into a policy document and adds the appropriate metadata for that content type. *script* Runs a custom JavaScript script from the Data Dictionary/Scripts folder. There are a number of sample scripts available. The list can vary depending on how Alfresco One is configured. **check-in**: Checks in files that are currently checked out. For example, files will be checked in before being moved to another folder. Select the option to indicate whether they will be checked in as minor or major versions. *transform and copy content* Action for transforming and copying content. You can add copies of files, in the format of your choice, to another location. For example, you can generate a copy of a Word document in PDF format in a different folder. *remove-features* Removes a property aspect from files to remove functionality or properties. *check-out*

Property	Description
	Checks out files automatically with a working copy created in the location of your choice. Select the option to associate a name or type with the file. *copy* Creates copies of files in the location of your choice. Set the additional deep-copy and overwrite-copy options to true if you want to copy or overwrite sub-folders and their contents. *transform-image* Action for transforming and copying image files in the format of your choice to another location. For example, you can generate a copy of GIF file in PNG format in a different folder.

2.8. Form editor

The form editor provides a powerful drag and drop interface to let you design forms from a rich set of controls. You can define form outcomes and create forms with multiple tabs. Individual controls and whole tabs can be made visible depending on the value of other form fields and process variables. You can design your form with groups of controls in varying numbers of columns.



In the example above, the form editor is open on a form containing two controls, a text box, and a multiline text box.

2.9. Business Rules - Decision Tables

There are many situations in a business process where you wish to evaluate some data you have collected and come to some conclusion or decision. Business rules provide a natural way to express the logic of decision making. Typical decision examples are calculating discounts, credit ratings, who to assign tasks to, what service level (SLA) to use, and so on.

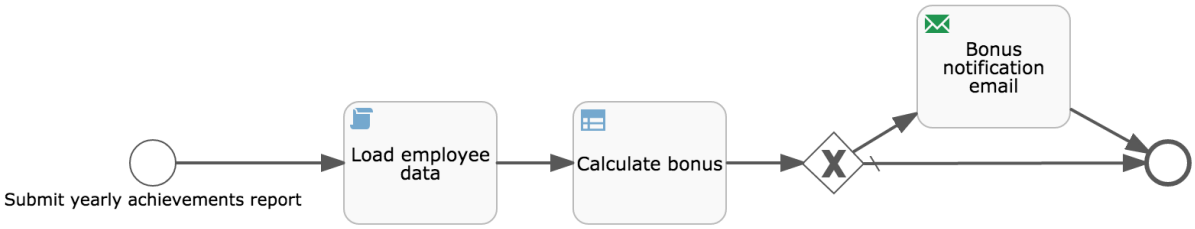
There are business rule systems that are hugely complex and intended for a wide range of uses. You can, of course, integrate Alfresco Activiti to these systems if they provide what you need. Often, within a business process, the rules can be very focused and need to be managed by business users. This is where Activiti's Decision Tables provide a natural solution.

In a Decision Table you only test, set and create variables using a set of business rules. There are no other side effects possible, such as calling out to external systems, because these are not needed: the Activiti process can do all this using the full range of its BPM capabilities before or after a Decision Table task.

You can think of a Decision Table as a spreadsheet that allows you to define a row for each business rule, with columns representing each variable that needs to be tested or set. There are two parts to a rule: the *conditions* (**if** they all match, the rule "succeeds") and the *conclusions* (**then** set some values). In each cell of the table there can be a value expression that is used to try and match against variable's values, or to calculate the value to set. When a Decision Table is evaluated, it tries all the rules in turn (so ordering of rules matters), testing and setting values. Depending on how you want the rules to be interpreted, you can set the rules to stop as soon as one rule matches and succeeds in setting its values, or to run through all the rules, setting values for every matching rule. If it runs through all rules, you can think of the last successful rule winning, as it may overwrite values that were set for the same variables in other successful rules.

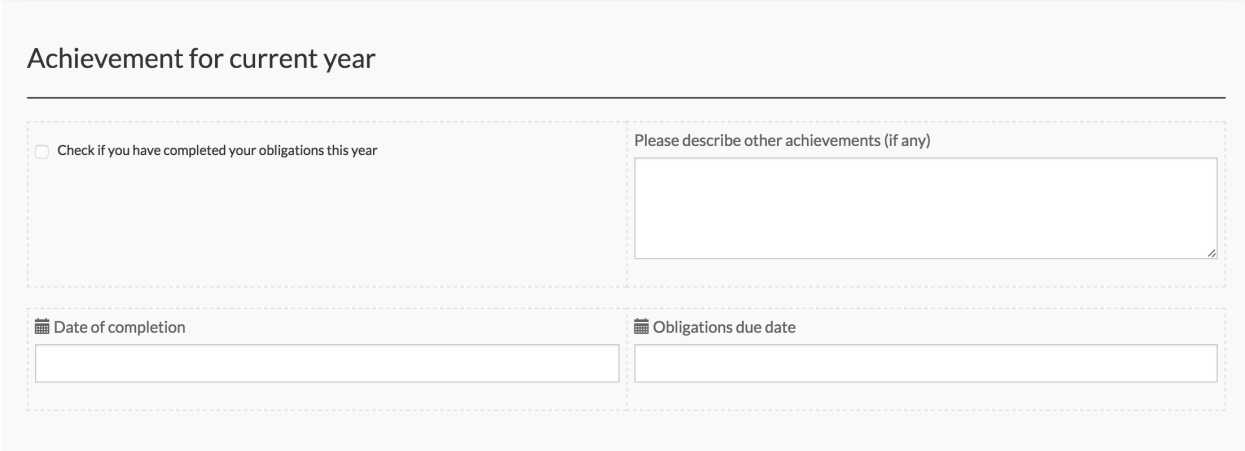
Activiti's Decision tables follow the Decision Model Notation (DMN) specification.

In the following, we will create a simple process that makes use of a Decision task and its Decision Table. We will use the BPMN editor, but you can just as well use the Step editor to achieve the same result. First let's take a look at the process we want to create:



In this "Annual Work Review" process, a user can enter the details of his or hers achievements for the current year, and if the work efforts have gone beyond the employee's obligations a bonus will be given and an email sent notifying the user about it. The logic to decide if a bonus should be given or not is implemented using a Decision Table in the "Calculate bonus" Decision task above. Before we take a look at the Decision Table itself, let's quickly take a look at the tasks before the "Calculate bonus" Decision task.

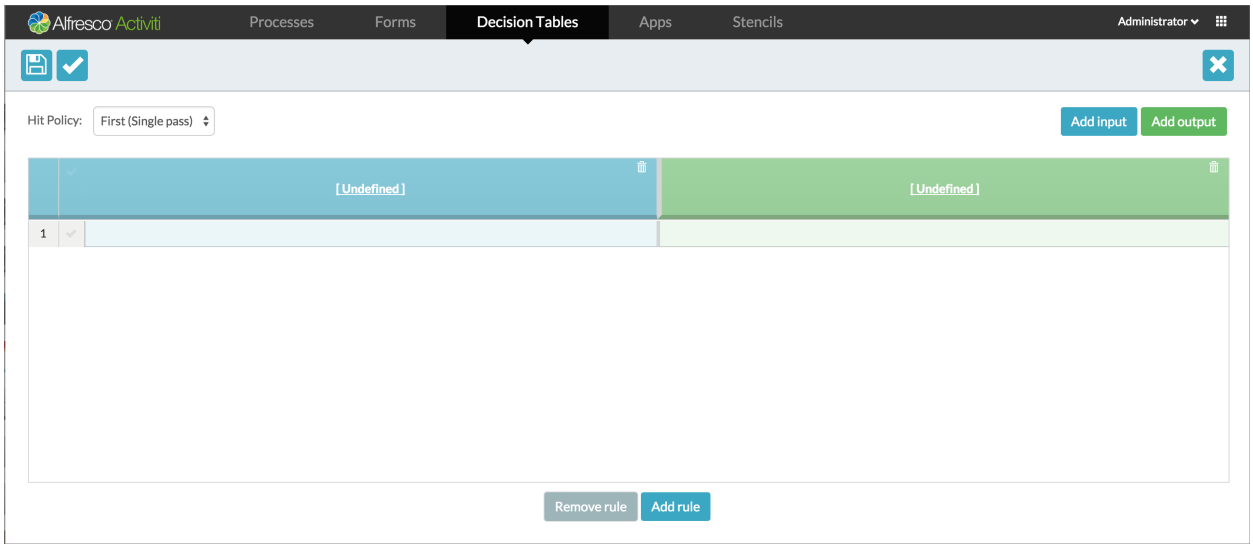
The process' start form is shown below and defines 4 fields: *obligationsCompleted* (*boolean*), *additionalAchievements* (*string*), *completedDate* (*date*) and *dueDate* (*date*). See the Form editor section for more information on how to create forms.



The second task in the process is a "Script task" that we are using to load some demo user data. It has format *javascript* and declares 2 variables: *yearsOfService* (*integer*) and *salary* (*integer*) in the "Variables" property dialog. In the "Script" property dialog for the script task the following code has been added to get some employee data.

```
execution.setVariable("salary", 1000);
execution.setVariable("yearsOfService", 5);
```

Now we are ready to create our Decision Table that will have all the input values it needs to decide if a bonus should be given or not. The decision task is created by dragging and dropping a "Decision Task" from the "Activities" section in the editor palette. The only mandatory property is the "Referenced decision table" property in which you should choose "New decision table". Enter "Calculate Bonus" as the name and click the "Create decision table" button to be taken to the decision table editor, as shown below.



Before starting to look at the details of the editor, let's start by looking at the rules that we want to create to decide if a bonus should be given or not. The logic (or the rules) we will create can be seen in the Decision Table below:

	Obligations completed? [obligationsCompleted]	Additional achievements [additionalAchievements]	Years of service [yearsOfService]	Completed date [completedDate]	Bonus [bonus]
1	== true	!= empty	> 5	< fn_subtractDate(dueDate,0,3,0)	salary * 0.05
2	== true	!= empty	-	-	salary * 0.03
3	== true	-	> 5	-	salary * 0.03
4	== true	-	-	< fn_subtractDate(dueDate,0,3,0)	salary * 0.03
5	-	-	-	-	0

The logic can be summarized as:

- IF the user has completed the obligations AND has performed additional achievements AND has worked for the company more than 5 years AND completed the obligations 3 months before the due date
    - THEN the bonus is 5% of the salary
- IF the user has completed the obligations AND has performed additional achievements
    - THEN the bonus is 3% of the salary
- IF the user has completed the obligations AND has worked for the company more than 5 years
    - THEN the bonus is 3% of the salary

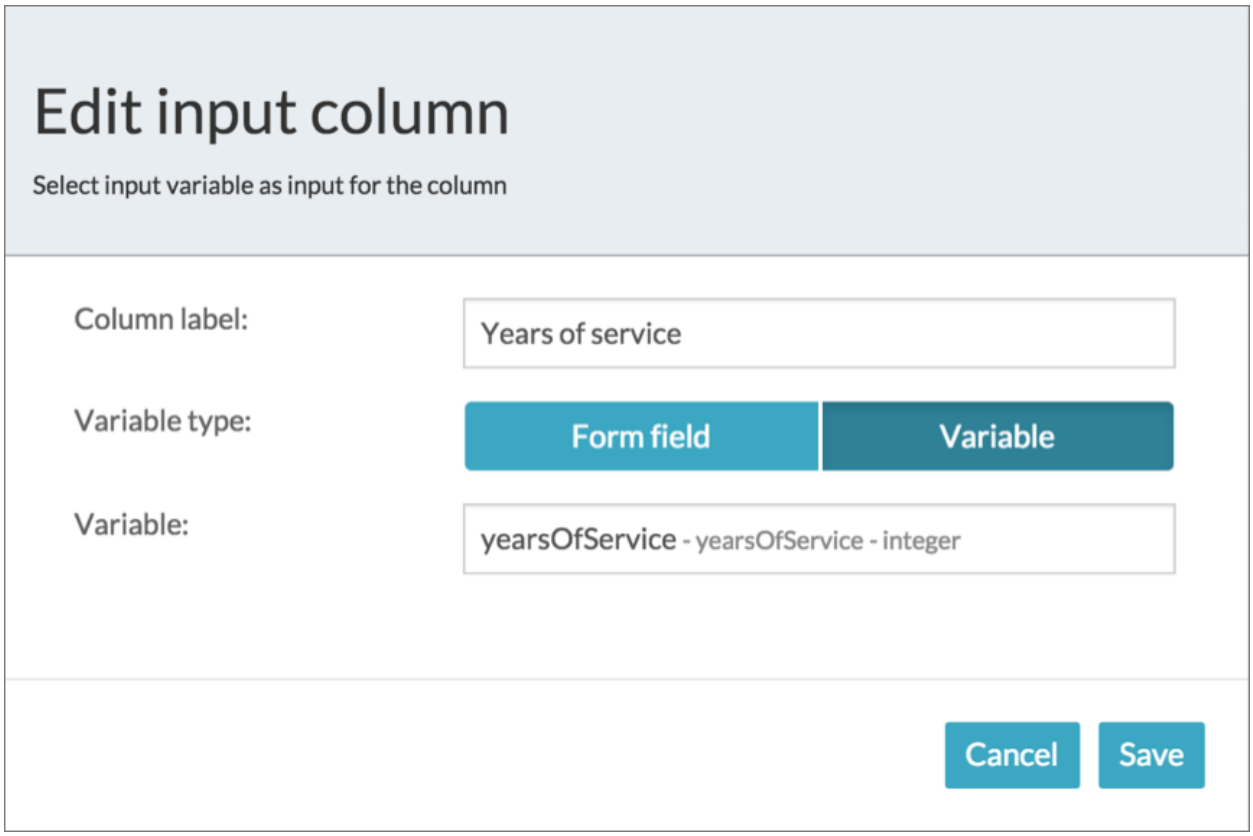
- IF the user has completed the obligations AND completed the obligations 3 months before the due date
    - THEN the bonus is 3% of the salary
- IF none of the rules above matched (empty cells are treated as an automatic match)
    - THEN the user gets no bonus

The expressions in each cell is an MVEL expression. MVEL is an embeddable scripting language that you can read more about here. Note though that you don't have to write MVEL syntax yourself but can use the edit icon in each cell to display a structured expression dialog where you can create these expressions through a simple interface. Once you are familiar with the syntax you can just enter them directly in the cells, like editing a spreadsheet.

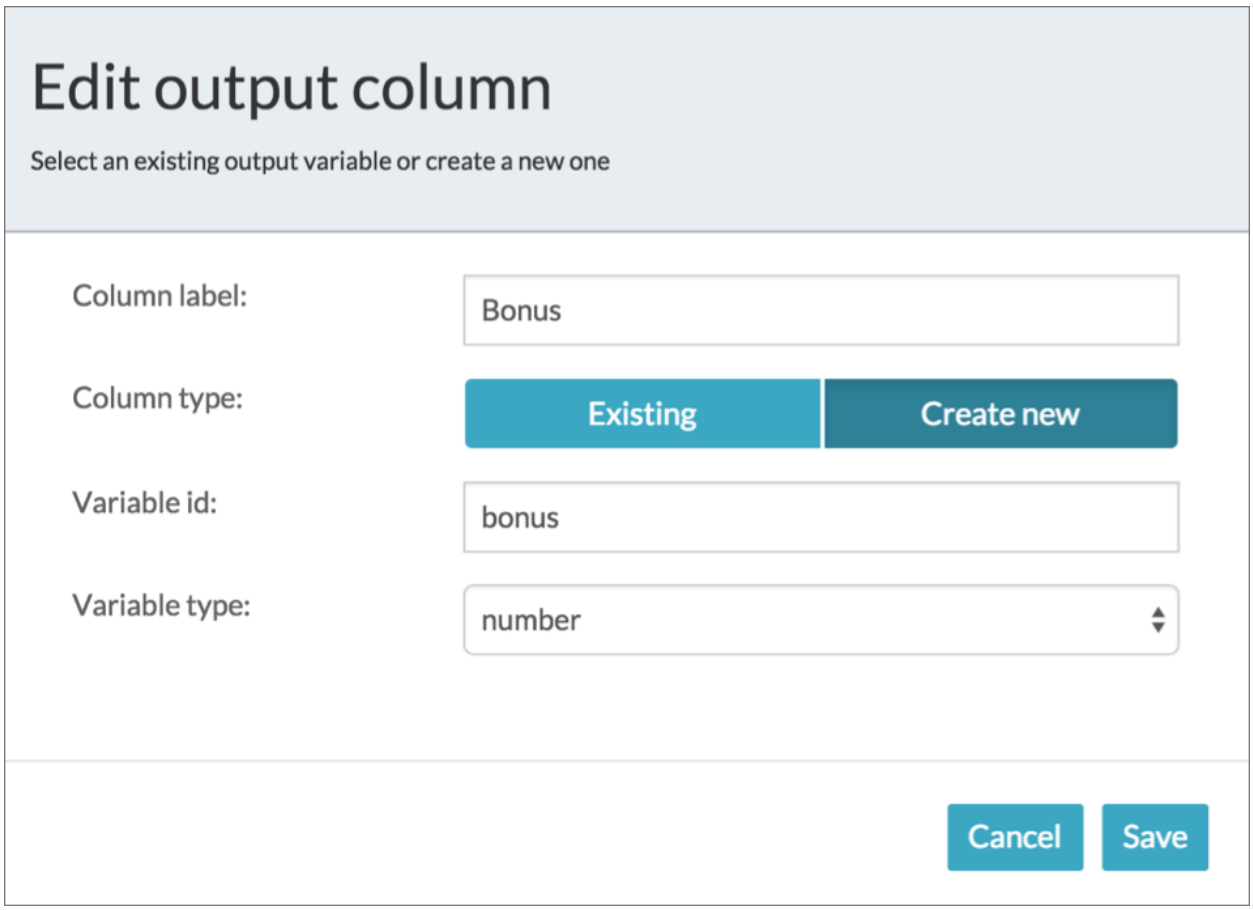
Note: By default it is NOT possible to type in any MVEL expression, but only a sub-set that is supported by the structured expression editor. If you want to be able to write more complex MVEL expressions, you may do so globally by your system administrator setting the *validator.editor.dmn.expression* property in *activiti-app.properties* to *false*. By disabling the default validation you will be able to run any MVEL expression, but you will not get the same help validating that your syntax is correct. Also, you will get a warning message in the structured editor when trying to open an expression it isn't able to recognize.

Even if you don't know MVEL, most expressions are self-explanatory. The complex date expression *< fn_subtractDate(dueDate,0,3,0)* probably requires a small explanation though. A custom Activiti calculation for dates is used that takes the dueDate as the first parameter and then will calculate a date value by subtracting from it the last 3 parameters for *years*, *months* and *days*. In this case the expression checks if the *completedDate* is 3 months before the due date.

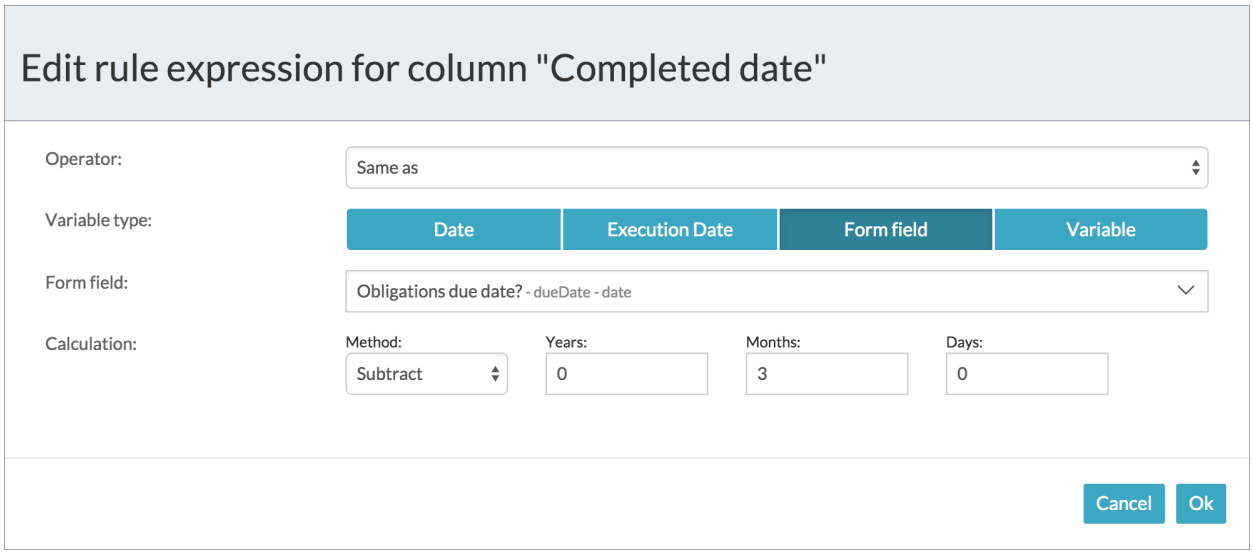
Now create the Decision Table above for yourself. The first thing you need to do is add 4 input expressions using the "Add input" button in the Decision Table editor. For each of these, select the process variable or form field to use as input for the column. When adding *yearsOfService* it should look like the following.



Then you need to add an output column by clicking "Add output", making sure the dialog looks as below to create a new process variable named *bonus*.



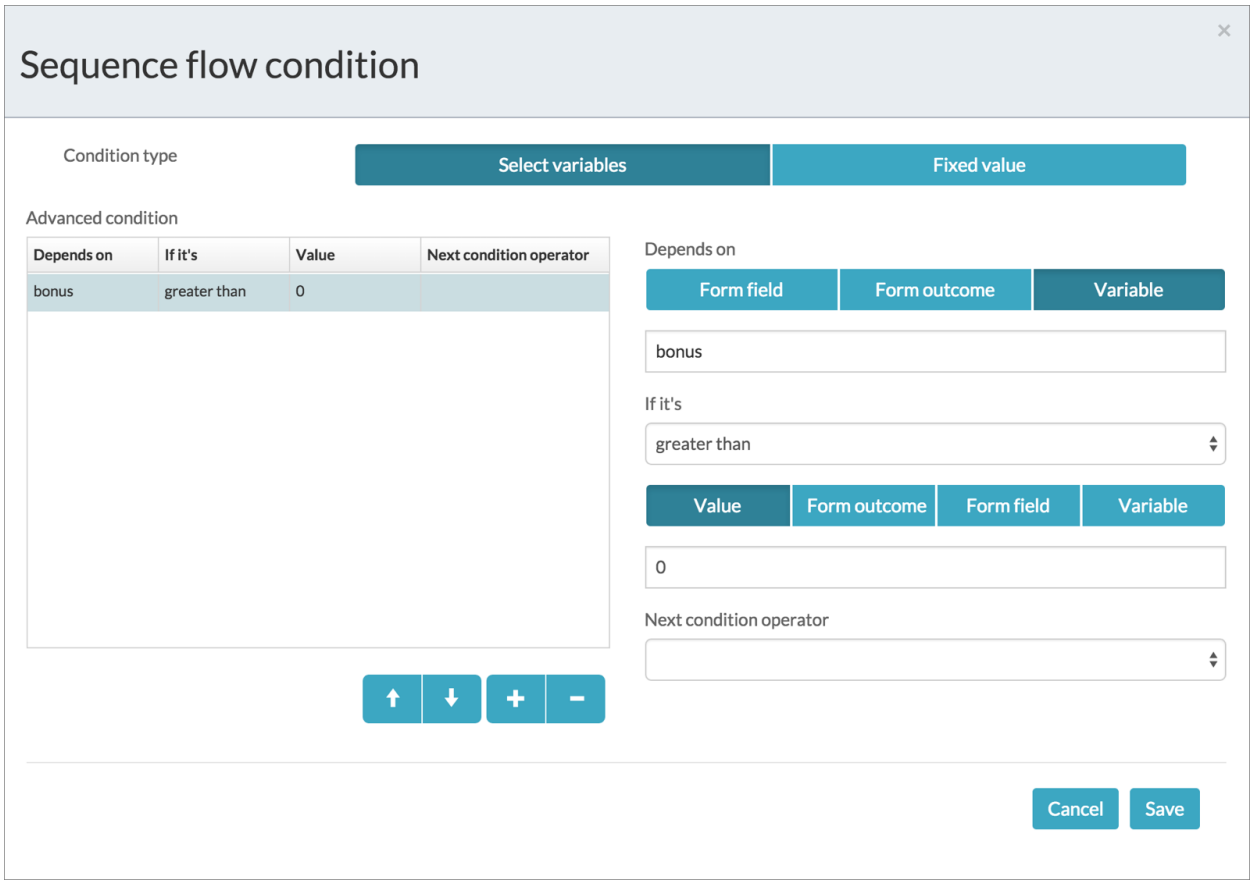
Time to add our rules. Feel free to type them directly into the cell or use the structured editor (which pops up when clicking the edit icon to the right in each cell). Below you can see how the structured editor looks like when adding the date expression from above.



When done, click **Validate** to make sure your decision table doesn't contain errors. Note that once you click **Validate**, the editor will validate your table for every change you make. When you're happy with your table, click the save icon. You will be prompted to give a "Decision Table key" which can be any value unique to the process.

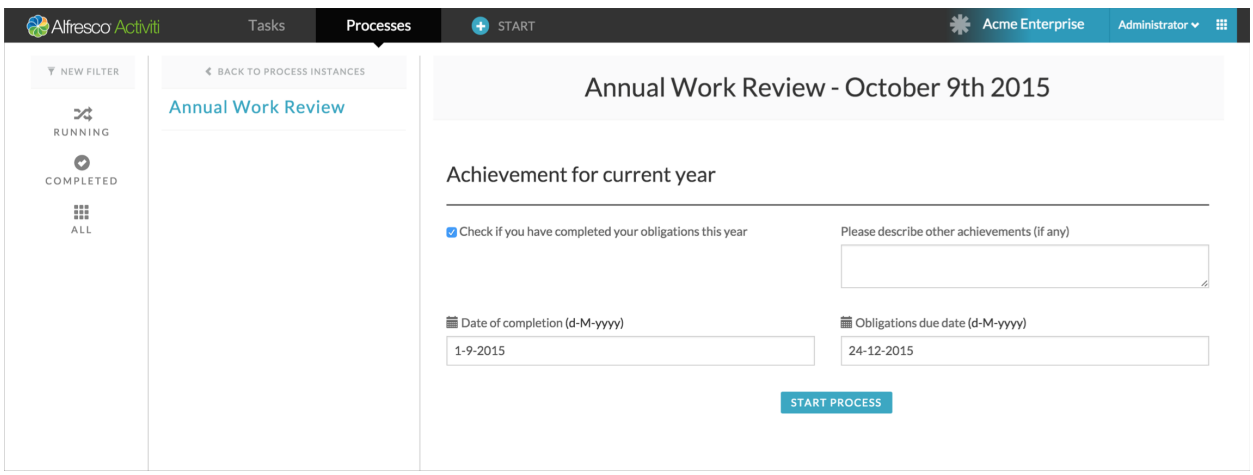
Back at the BPMN editor add an "Exclusive gateway" and from it add a new "End event" by clicking the circle with the thick border. Select the arrow that connects them and enable the "Default flow" property.

Now drag and drop a "Mail task" and set its "To" property's "Fixed value" to *${emailBean.getProcessInitiator(execution)}* so it sends the email to the initiator of the process. Then enter values for its "Subject" and "Text" (or "Html") properties. Add a sequence flow arrow to connect the gateway to the email task and make sure to set its "Flow condition" property to have an advanced condition as in the image below.

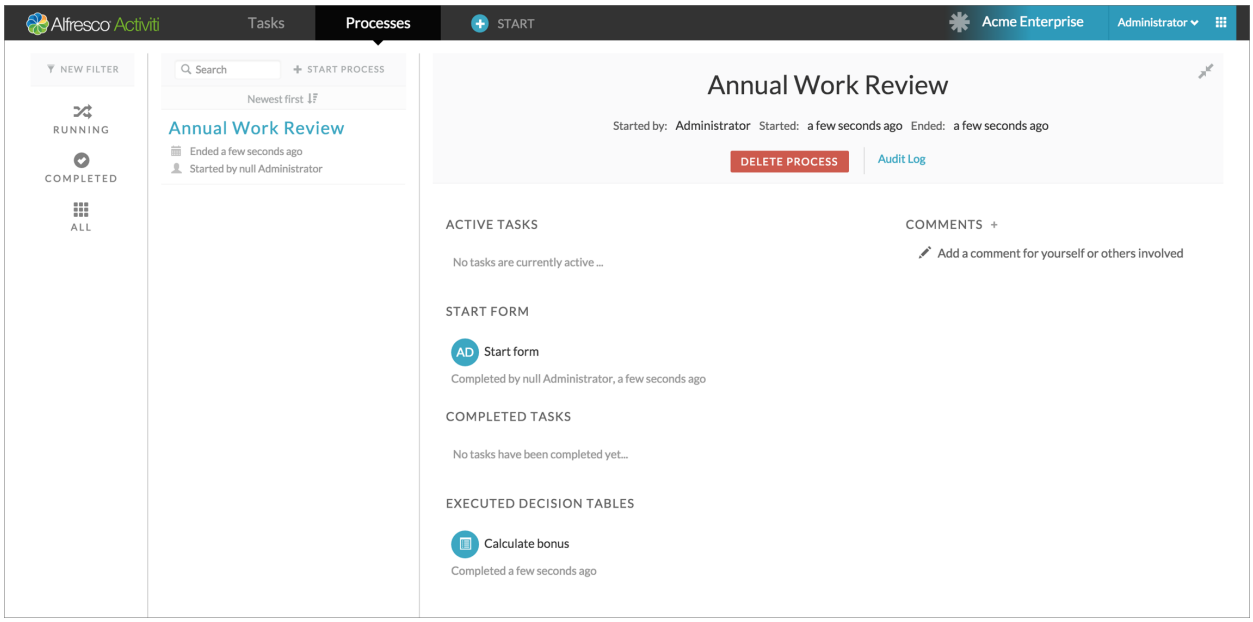


Finally, draw the sequence flow arrow from the mail task to the end event.

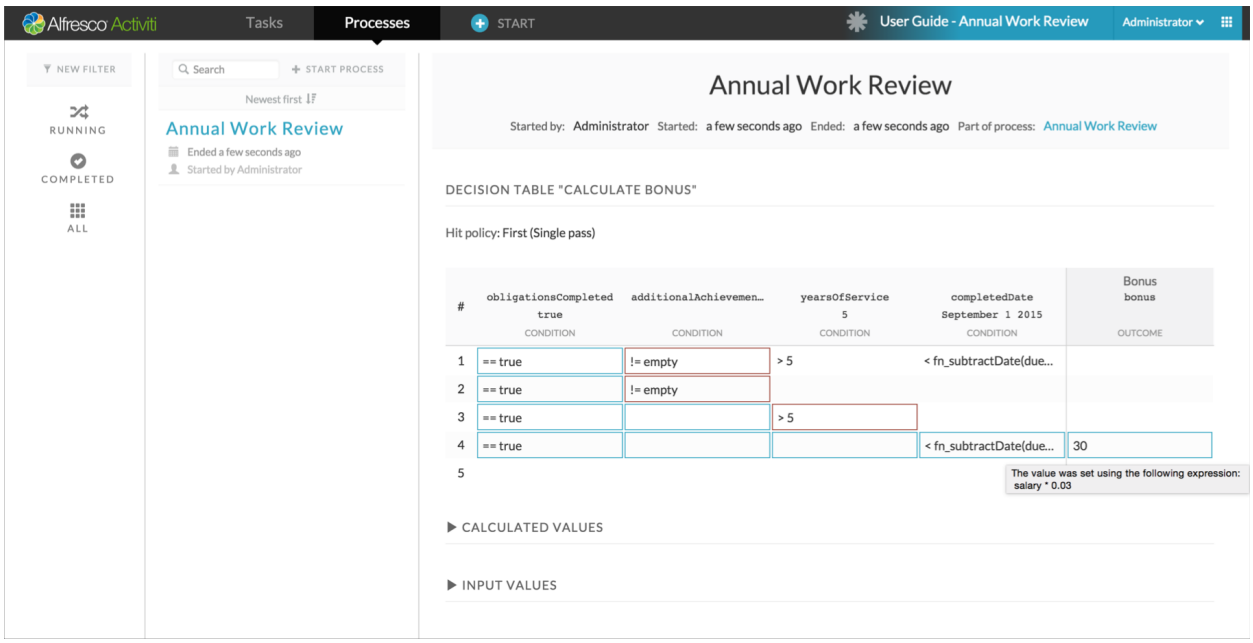
We are now ready to use our Decision Table in the Task app. Once you have deployed your process, start an Annual Work Review process by entering the following details in its Start form and click **Start Process**.



The process detail view is displayed as shown below. After a decision table is executed in a process, it is listed in the Executed Decision Tables section. If something caused the decision table to fail during execution, a red icon with a message is displayed stating an error occurred. Click the **Calculate Bonus** decision table in the user interface to see details about the decision table and its evaluation.



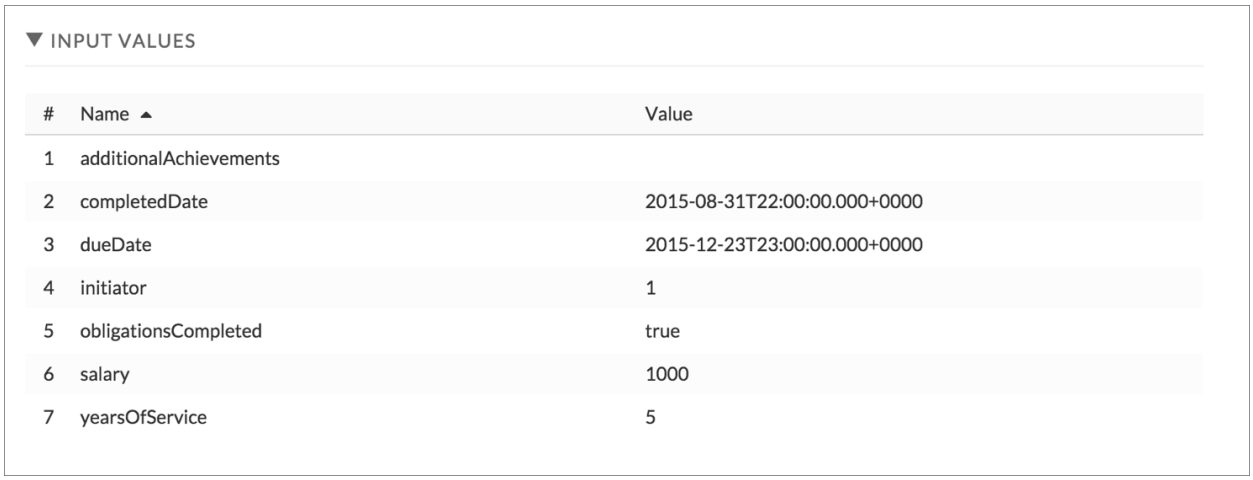
A decision table is a bit like a black box. You can see the history of it when it was executed. In the image below you can see the audit trail of the decision table.



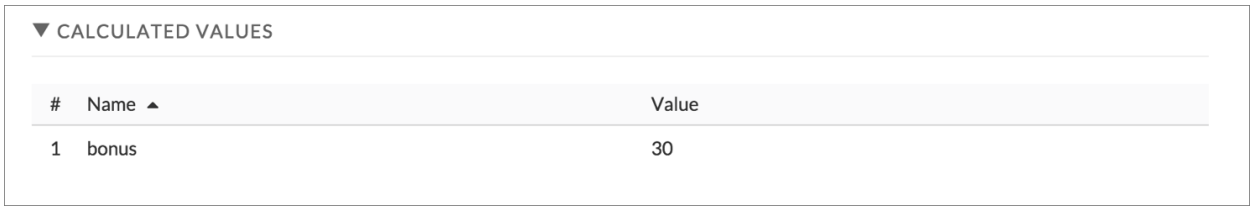
An input cell marked with a blue border indicates that the expression in the cell matched the input value. If a cell border is red it means it did not match. If it has no border it means it wasn't evaluated at all (for example, a previous cell had failed to match and is shown as red). If an exception occurs during evaluation it is also marked with a red border, but also with a red error icon in the right part of the cell.

An output cell only displays the value that was set by its expression. A blue border indicates that it was successfully set. A red border indicates an error occurred during execution of the cell expression. For tooltip information, position your cursor over a cell. An example of this can be seen in the image above where the output cell sets the bonus to "30": hover over the cell and the expression used to calculate the value is displayed.

To see a list of all the input values that were provided to the decision table before execution, click the "Input values" section and you will see the table below.



To see a list of all the output values that were set by the decision table after execution, click the "Output values" section and you will see the table below.



You may have noticed that we haven't yet mentioned anything about the decision table's "Hit policy". The hit policy decides "how" the decision table will be executed when rules succeed (a "hit"). In our decision table we have selected "First (single pass)", which means the decision engine will execute all rules in the given order until it has found a rule where all cell expressions match their input values. Then no further rules will be tested and the outcome expressions specified on the successful rule will be used to set the output values.

Empty cells are considered to be an automatic match, meaning that a rule with only empty cells will always be treated as succeeding (a hit). In our decision table we have such a rule in row #5, but with the input we gave, it will find a match on row #4 and the rule on row #5 will never get tested.

If we change the Hit policy in our table to be "Any (single pass)" the result after executing the decision table will be different. The execution evaluate all rows until the last rule, even if it found a rule that matched on a previous row.

Given the rules in our example, the *Any* hit policy does not make much sense, since the result would always be that bonus is set to "0" because the last rule always matches, no matter what input is given.

2.10. Data Models

A Data Model enables you to access and manipulate data related to a business process in Activiti. For example, you can define a data model that maps to a relational database (via JDBC) or a custom API to connect to an external source such as a patient database or a customer database.

To use the Data Model functionality effectively, perform one or all of the following steps:

- Reference an entity while mapping variables.
- Make entity fields visible in the process by mapping them.
- Reference mapped entity fields in forms when creating or editing forms.
- Reference entity fields in expressions when creating or maintaining decision tables.

2.10.1. Connecting your data model to a relational database

You can establish connection from your process in Activiti with a relational database. To enable the connection, you must first register the data source for your tenant in the Identity Management app in Activiti.

Configuring the data source

Before defining a data model, you must establish a database connection and register the data source in your tenant.

To configure the data source:

1. In the **Identity Management** app, click **Tenants** > **Data sources**.
2. Click + (plus icon) and configure the following settings (see the *activiti-app.properties* file for more details):
    - **Name** – Name of your data source. For example, modeler.
    - **JDBC url** – The JDBC URL used to connect to the database. For example:

      jdbc:mysql://127.0.0.1:3306/modeler?characterEncoding=UTF-8
    - **Driver class** – The JDBC driver used to connect to the database. For example: com.mysql.jdbc.Driver
    - **Username & Password** – The username and password of the account used to connect to the database.
3. Click **Save**.

2.10.2. Defining data models

Once defined, Data Models enable you to read, insert, update, and delete entities while working through your process.

To define a data model:

1. From the **Kickstart** app, click **Data Models**. The Data Models page is displayed.
2. Click **Create Data Model**. The Create a new data model dialog box appears. Or to import an existing data model, click **Import Data Model**.
3. Select the data source that you defined in Identity Management.
4. Click **Add Entity** and enter data in the following fields:
    - **Entity name** – The name you want to use for the entity, for example, Customer.
    - **Entity description (optional)** – Description of the entity.
    - **Table name** – The database table name that you want the entity to be mapped to, for example Customer.
    - **Attributes** – Displays the entity attributes as you add them.
5. Click **Add Attribute** and enter data in the following fields:
    - **Attribute name** – Name you want to use for the attribute, for example, Customer Id.
    - **Attribute description (optional)** – Description of the attribute.
    - **Column name** – Column name as specified in the database, for example, id.
    - **Attribute type** – One of the following attribute types: String, number, date.
    - **Primary key** – Select to indicate if the attribute is a primary key or not.
    - **Database generated value (autoincrement)** - Select this if the primary key is set to autoincrement in the database.
    - **Required** – Select to indicate if the attribute should be mandatory or not.
6. Save the data model.

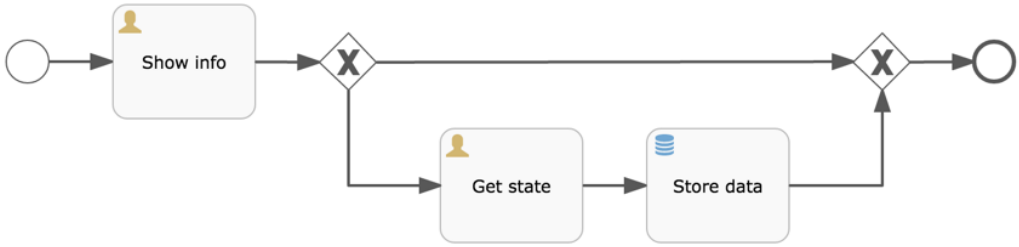
Note: The **Remove entity** and **Remove attribute** buttons can be used to remove entities and attributes respectively.

2.10.3. Using data model in your processes

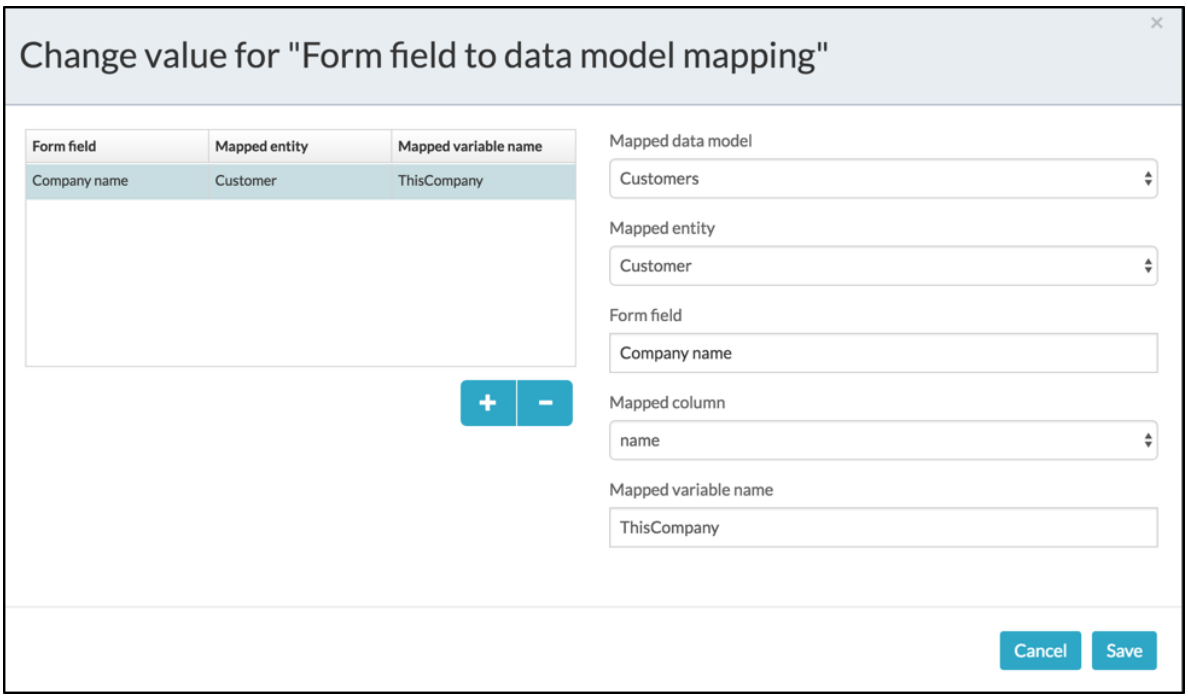
Once you have defined the data model for a database data source, the next step is to use them in forms, decision tables, and process conditions, by mapping them into form fields or process variables. For example, to use patients' information, you can map their information such as their name and address into your forms.

To start accessing data using your data model:

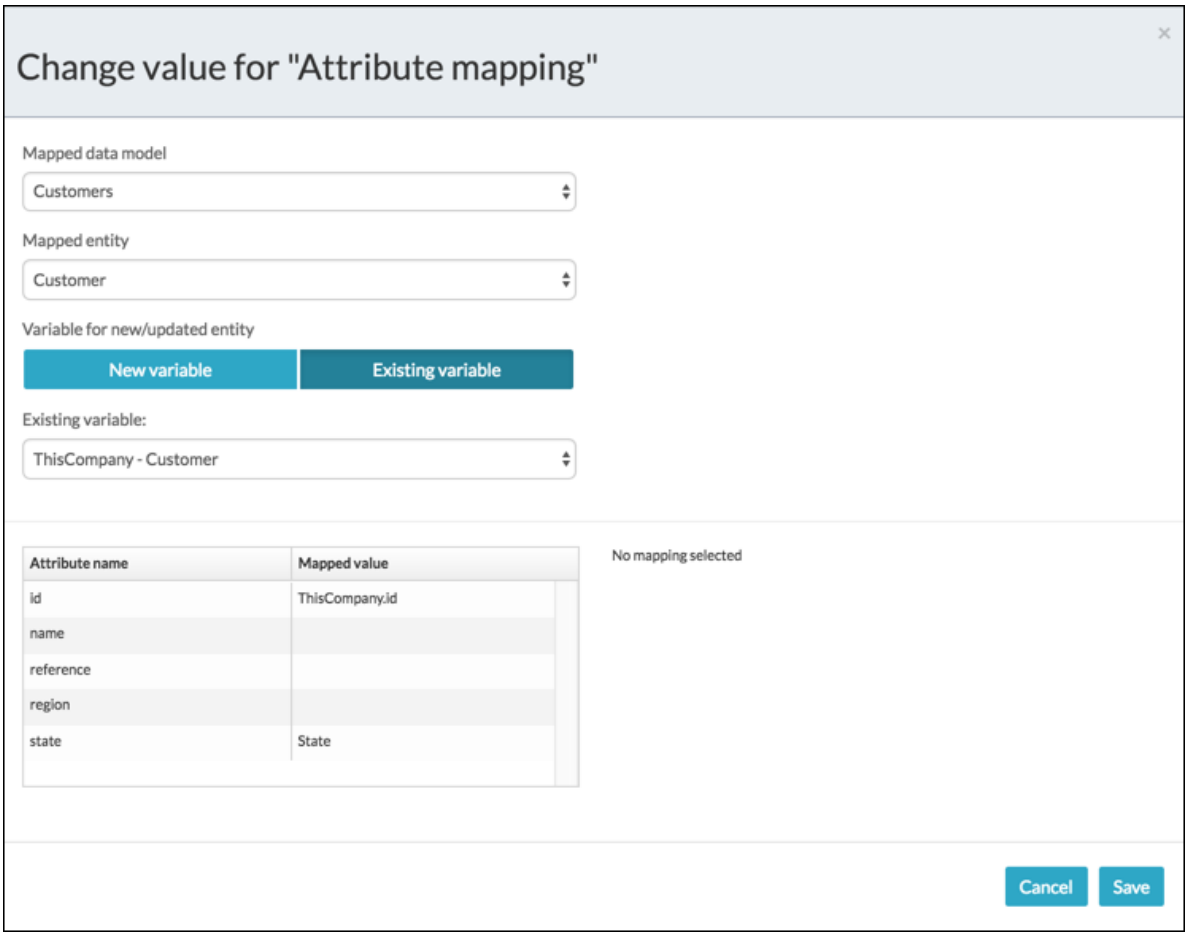
1. From the Kickstart app, create a simple business process model with a BPMN task that includes a Start event, Store entity task, and an End task.



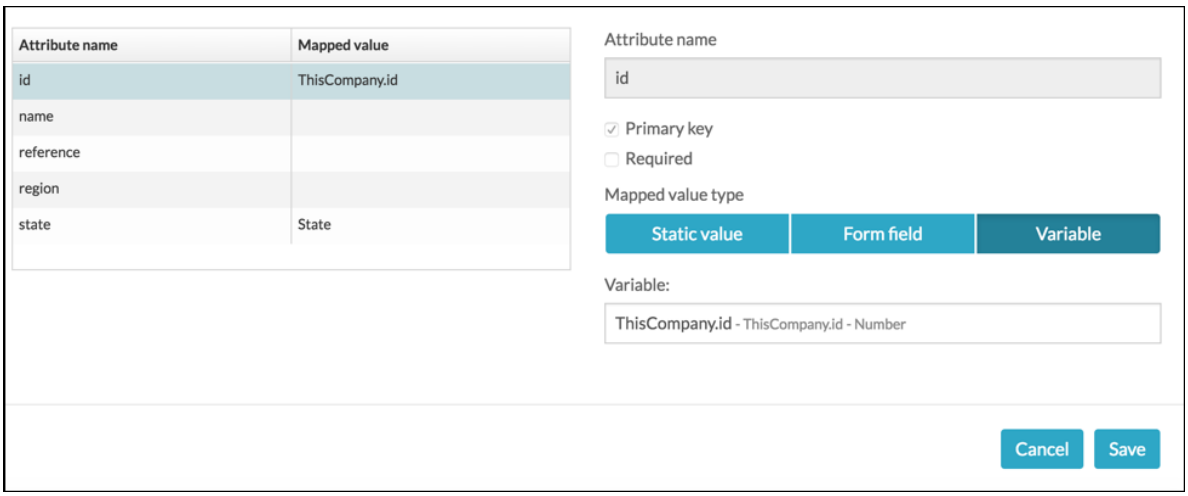
2. From the BPMN editor, select the Start event and then click **Referenced Form** to select an existing form, or create a new form. The Form reference dialog box appears.
3. Select the form that you want to customize and click **Open**.
4. In the selected form, drag a text type field from the palette, rename it as Company name and then save it.
5. From the BPMN editor, click **Form field to data model mapping**. A dialog box to change value for form field to data mapping appears.
6. Map the fields for Company Name as shown below:



7. From the BPMN editor, click the **Store entity task** and then **Attribute mapping** to edit the mappings. The Change value for Attribute mapping dialog box appears.



8. Select the **Mapped data model** and **Mapped entity**.
9. Add a new variable or use an existing variable. In this case, select an existing one: *ThisCompany — Customer*.
10. Map the attribute names with mapped value by selecting the required attribute in the Attribute table as shown below:



11. Configure the variable for the selected mapping, and then click **Save**.
12. Publish your app and verify the data connection by making changes to the process data.

2.10.4. Saving data using your data model

As you collect new data about an entity, you may wish to save this back to the database. However, as this is not done automatically when a form is saved, you must create a task in your process to explicitly save the data you want.

To save data using the data model:

1. From the **Kickstart** app, edit the business process model you created above to access data.
2. In the **Visual Editor**, drag the **Store entity task** activity type from the palette and place before the process **End**.
3. Remove the link from the selected task and connect the Store task between it and the process End.
4. Edit the Store entity task activity and click the attribute mapping field. Configure the following settings in the Attribute Mapping dialog box:
    - **Mapped data model** – Select the data model to map your entity with.
    - **Mapped entity** – Select the entity to map your data model with.
    - **New Variable/ existing variable** – Create a new variable or select an existing variable.
    - **Attribute name** – Map the attribute names with the relevant form fields by selecting the relevant form field value from the drop-down list. For example, Customer Id with ID and Customer name with Name.
    - **Mapped value type** – Select one of the value types for mapping attributes. In the above example, Form field was selected. However, you can also map your attributes with a static field or variable.
5. Create a new app definition and associate your process with it.
6. Deploy the app and test it by updating the data. For example:
    - Open your app and click **+ START**. The form fields that you defined in your process appear.
    - Edit an existing Id (column name) with a new customer name and verify if the changes appear in your database.

Sample database table

While working on the data model functionality, locate or create a database table and its columns from your database and make sure to create matching attributes in your Data Model. For example, the following customer table was used for the customer data model in the above sections.

Customer table			
Attribute name	**Type**	**Data Type/Size**	**Constraints**
id	Number	INT (11)	Primary key, Auto Increment
name	String	VARCHAR(255)	
region	String	VARCHAR (20)	
state	String	VARCHAR (45)	

2.11. Creating your first process

You create an Alfresco Activiti process model to represent a series of tasks in your business process. This tutorial guides you through creating a simple process model.

The process you are modeling here is a simplified business project lifecycle. Each project has a name, type, due date, and documents associated with it. Each project is started, and then reviewed to determine if it should be accepted on to the project list, or rejected.

1. From your landing page, click the **Kickstart** app tile.
2. Click the **Create Process** Button.

   The Create a new business process model dialog appears.
3. Give your new process model a name.

   For example, **First Process**.
4. In the Editor type drop-down, select the Step editor.
5. Click **Create new model**.

   The Step editor is displayed.

   The first step, Process start, is already added to your process. You are going to set the process to start by having the user complete a form.

6. Click on the Process start step.

   It expands to allow you to change the step.

   If you have some forms in your Forms library, they will be listed here, and you can pick one, but in this tutorial we will create a new form.

7. Click on the Start form box.

8. Click **Create form**.

   The Create a new form dialog appears. The form you create now is part of this process model and is not available in your forms library for use in other process models. If you want to create a form you can reuse in other process models you can do so from the Forms page.

9. Give the new form a name.

   For example, `Start form`.

10. Click **Create new form**.

    The Form Editor is displayed. Design the form by dragging and dropping the field types from the palette to the Form Editor. You can hover over each field in the Design area, and click the pencil icon to edit the field properties, or to remove the field from the form. Each field type offers different options. You can also add a display label in the process to reference a value entered in a field by a user in a running process. You can also define if the field is mandatory for the form to be completed. In this tutorial, you just give labels to the fields.

11. Drag and drop the required fields from the palette to the canvas on the right-hand side.

    From the screen shot you can see your form has four fields. .. A Text field for the project name. .. A Date field for the project's start date. .. A group of three Radio buttons to select the project type. .. An Attach control to allow the user to store project documents.

12. Click the **Save** icon to save your form.

    You are back in the Step editor.

13. Clicking the + icon below the Process start box to add the first step in your process.

    You need to add a Human step that can be used to assign a task to a user.

14. Select the Human step and fill in a name in the step box just created.

For this tutorial, use the name `Review project`.

The Human step allows you to select who the task should be assigned to. You can assign the person who initiated the process, a single named user, a set of candidate users, or depending on the type of your account, a group of users. When a task is assigned to a group or a list of candidate users, all of those users can see the task in their task list, and will need to claim it in order to complete the task. For this tutorial, you will assign all tasks to the process initiator, that's you, so you can run the process and see the tasks yourself.

15. You need to create a form to allow review comments before we create the next step in the process.

    a. On the Form tab for the Review project step, create a new form called `decide`.

    b. Add a Multiline text field and name it **Review comment**.

    c. Select the Outcomes tab and choose the Use custom outcomes for this form option, and add two outcomes: Accept and Reject.

    d. Save the form, and return to the step editor.

16. Add a Choice step by clicking the + icon below the Review Project step.

    This step allows you to take a different action depending on the outcome selected in the associated form.

    You can add more choices by clicking on the + icon in the middle of the Choice step. For this tutorial, we only need two based on your accept and reject outcomes..

17. Click on the First choice box. A dialog allows you to select a condition based on existing form fields or outcomes. For this tutorial, set the First choice to a Form outcome. Choose the `decide` form from the drop-down list of those already added to the process, and then select it to be Equals, to the value Accept.

18. Click on the Second choice box, and repeat the last step, this time choosing the value Reject.

19. You need to add a task to be done if a project review is accepted.

    a. Under the First choice click the + icon.

    b. Add a Human step with the name `Update project list`.

20. You need to add a task to be done if a project review is rejected.

    a. Under the Second choice click the + icon.

b. Add a Human step with the name `Inform project leader of rejection`.

c. Since the process should stop after rejection, add an End process step under the `Inform project leader of rejection` step.

21. Add a final step after the accept/reject choice step to display the project details by clicking the + icon at the bottom of the step diagram..

    a. Add a Human step with the name `Show Project Details`.

    b. On the Form tab for this step, create a new form. Drag a Display text field to the canvas, and enter a text message to display.

       The text can contain references to values for forms in the process. There is a helper drop-down list from which a form field reference can be selected. It is inserted at the current cursor position in the text.

    c. Save the form.

       The step editor is displayed.

22. Save your completed process model.

    Your process is listed in the Process tab as a thumbnail of the process. You can edit any process from the list by clicking the BPMN Editor button in the top right corner of the thumbnail. You can see additional information about a model by clicking on the thumbnail itself or the Show Details button in the top right corner of the thumbnail. This takes you to the Details page for the process model. Here, you can see a read-only preview of the model and the actions you can perform on it.

Now that you have created a process, you need to create your first app so you can publish and deploy your process model.

2.12. Creating your first app

You create an Alfresco Activiti process app to group together a number of processes to make them available to yourself or other users. An app is the container for handling a group of published processes and deploying them to an Activiti engine. This tutorial leads you through the steps required to create and use an app containing a single process.

This tutorial uses the process model created in the Creating your first process tutorial.

1. Go to the **Kickstart App** > **Apps** tab, and click **Create App**.

   The Create a new app definition dialog appears.

2. Enter a name for your app and click **Create a new app definition**.

   Use the name **My First App** for this tutorial.

3. Choose an icon and theme for your new app's tile.

4. Click **Edit included models** and select a process(s) of your choice. In this case, select the published model from the Creating your first process tutorial.

5. Click **Save**, and select the **Publish** check box in the Save app definition dialog to save and publish your app.

   Publishing an app makes it available to everyone you've shared it with.

6. You can now add the app as a tile on your landing page.

   a. On the landing page, click on the last tile labeled with a + (plus) icon. The Add app to landing page dialog appears.

   b. Select **My First App** from the list of published apps and click **Deploy**.

      A new app tile is added to your landing page.

Your app is now deployed and ready to be used.

2.13. Starting your first process

You start a process from the Processes tab of the Task app page. In this section, you are going to start and monitor the process you designed in the previous tutorial. To start the process, first add a process to an app and deploy that app deployed.

The following steps use the process model created in the Creating your first process tutorial, and the corresponding app created and deployed in the Creating your first app tutorial.

1. Go to the **Task App** > **Processes** tab, and click **Start**. button

   The form you created in the Creating your first process tutorial is displayed.

2. Fill in the details on the form, and add any documents you need, and click **START PROCESS**.

   You are returned to the Processes page, which displays the process list with the process that you just started.

   On the Processes page, you can view running processes and see the current and completed tasks. You can also add comments that are available for anyone involved in the process.

3. Now that you have started the process, you should complete the tasks you defined in it. Go to the **Tasks** tab.

   The first step in the process is a task to review the project, and accept or reject it. Remember that when you created the first step in Step Editor, you specified that the task should be assigned to the process initiator. Since you started the process, you are the process initiator and this task is assigned to you.

4. Click **Show Details** or **Show Form**.

   At this stage you can add people, documents, and comments to the task.

5. Click **Show Form** to return to the form and then **Accept**.

   The Review Project task is complete and a new task, Update Project List is displayed. You defined this as a choice step in Step Editor, if the user choice was to accept the project.

6. Click **Complete** to go to the next step.

   The task that shows the details of the accepted project is displayed.

7. Click **Complete**.

   You have now completed all the tasks in the process and there are no tasks displayed for you in the Tasks tab. Now, if you click on the Processes tab, you'll not see any running processes.

You have started your first process, performed the tasks assigned to you in that process, and completed a process successfully.

2.14. Creating a single task

As you have seen from previous sections, processes are made up of individual tasks. You can also create a single task for yourself or others and assign it for completion. This tutorial guides you through the steps for creating and completing a single task.

In this tutorial you will add a single task `Brush teeth` and complete the task yourself.

1. On the Tasks tab of the Task app page click the CREATE TASK button

   The New task dialog appears.

2. Give your new task a name, and optionally a description, and click Create.

   Your new task appears in the task list, and the task details are displayed in the right-hand panel.

Now you have created a task you can alter the details such as the assignee and the Due date, involve others in the task, add a document and add comments to be shared with other collaborators in the task. For this simple task of `Brushing teeth`, you are just going to add a due date of `today`.

3. Click **Due date**.

   A date chooser drops down.

4. Click **Due today**.

   The Due date now has a timer displayed showing the number of hours before the end of the day. Many fields displayed in the Activiti app can accept user input when you click on them. The Assignee field the task is another example.

5. When you've brushed your teeth, click Complete in the task details area.

   The task is removed from the open task list.

6. By default, your task list displays only open tasks. That is why you no longer see the task you just completed. For completed tasks click the COMPLETED TASKS filter in the right-most column of the Tasks tab.

You have created and completed your first single task and used some of the filtering capabilities of the Task app.

3. Disclaimer