

## User task

In this session we will go through user task.

A User Task is used to model work that needs to be done by a human actor. When the process execution arrives at such a User Task, a new task is created in the task list of the user(s) or group(s) assigned to that task.



### User Task Properties:

#### General:

General Forms Listeners Input/Output Extensions

**General**

**Id**  
userTask

**Name**  
User Task

**Details**

**Assignee**  
demo

**Candidate Users**

**Candidate Groups**

**Due Date**  
  
The due date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

**Follow Up Date**  
  
The follow up date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

**Priority**

**Asynchronous Continuations**

Asynchronous Before  
 Asynchronous After

**Documentation**

Element Documentation

**Id:** Id of task

**Name:** Name of task

**Assignee:** This custom extension allows direct assignment of a User Task to a given user.

**Candidate Users:** This custom extension allows you to make a user a candidate for a task.

**Candidate Groups:** This custom extension allows you to make a group a candidate for a task.

**Due Date:** Each task has a field indicating the due date of that task. The Query API can be used to query for tasks that are due on, before or after a certain date.

The due date of a task can also be altered using the TaskService or in TaskListeners using the passed DelegateTask.

**Follow Up Date:** Each task has a field indicating the follow up date of that task. The Query API can be used to query for tasks that need to be followed up on, before or after a certain date.

**Priority:** The attribute specifies the initial priority of a User Task when it is created.

Default value is valid java long value : 50

**Asynchronous Before:** Specifies an asynchronous continuation after an activity

**Asynchronous After:** Specifies an asynchronous continuation before an activity

**Element Documentation:** A User Task can also have a description. In fact, any BPMN 2.0 element can have a description. A description is defined by adding the documentation element.

**Exercise User Task with Assignee:**



## User Task Properties:

**userTask**

**General**

Id	userTask	x
Name	User Task	/

**Details**

Assignee

demo	x
------	---

This user task is assigned to demo user.

## Task Form:

## userTask

General Forms Listeners Input/Output Extensions

### Forms

#### Form Key

#### Form Fields

x	+
firstname	
lastname	

#### Form Field

##### ID

 x

##### Type

 ▼

##### Label

 x

##### Default Value

When user task is created. Default form with field firstname and lastname will be presented to user demo.

## Deploy and Test the process

The screenshot shows a dialog box titled "Start process". At the top right is a search bar with placeholder text "Search by process name.". Below the title, there is a note: "Click on the process to start." followed by a small info icon. A list of processes is displayed: "Invoice Receipt", "Loan Approval Process", "Process Script Result Variable", and "Process User Task Demo". At the bottom right of the dialog is a "Close" button.

### Start Process:

Start Process User Task Demo.

### User task:

User task will be created and assigned to demo user as below

The screenshot shows a "User Task" form for "Process User Task Demo". The form includes fields for "Set follow-up date", "Set due date", "Add groups", and a user selection for "Demo Demo". There are tabs for "Form" (selected), "History", "Diagram", and "Description". The "First name" field contains "sumit" and the "Last name" field contains "Vadaviya". At the bottom are "Save" and "Complete" buttons.

Click the Complete button.

Process will be completed.

## **Exercise User Task with Candidate User:**

### **User Task Properties:**

Task\_14qmcj3

General	Forms	Listeners	Input/Output	Extensions
<b>General</b>				
Id	Task_14qmcj3 <input type="button" value="x"/>			
Name	User Task <input type="button" value="//"/>			
<b>Details</b>				
Assignee	<input type="text"/>			
Candidate Users	demo, john <input type="button" value="x"/>			
Candidate Groups	<input type="text"/>			

Demo and john user will be candidate for this user task to work on.

### **Deploy the process**

Login as demo user to Camunda TaskList application.

### **Create Filter:**

Create filter to list user task which demo user can work on.

## Edit filter

General

*Customize the appearance of the filter.*

Name	Task by Candidate user	Color	<input type="color"/>
Description	Shows the tasks I am assigned	Priority	0
<input type="checkbox"/> Auto-Refresh			

Criteria

Permissions

Variables

## Edit filter

General

Criteria

*This section is aimed to set the parameters used to filter the tasks. Keys marked with a \* accept expressions as value.*

Add criter...	Key	Value
Rem... <input type="button" value="x"/>	Candidate User *	<input type="text" value="demo"/> <small>E.g.: \${currentUser()}</small>

Include assigned tasks  
Also include tasks that are assigned to users in candidate queries.  
Default is to only include tasks that are not assigned to any user if you query by candidate user or group(s).  
 ALL of the criteria are met.

Permissions

Variables

[Delete filter](#) [Close](#) [Save](#)

Save the filter configuration.

## Start the Process

The screenshot shows a user interface for starting a process. At the top left is a button labeled "Start process". To its right is a search bar with the placeholder text "Search by process name.". Below this, a note says "Click on the process to start." followed by a small info icon. A list of processes is shown, with "Process User Task Candidate User Demo" being the first item. At the bottom right of the interface is a "Close" button.

## Claim and complete user task:

The screenshot shows the Camunda Tasklist interface. On the left is a sidebar with navigation links: "Create a filter +", "My Tasks", "My Group Tasks", "Accounting", "John's Tasks", "Mary's Tasks", "Peter's Tasks", "Task by Candidate user (1)", and "All Tasks". The main area is titled "User Task" and displays a single task: "Process User Task Candidate User Demo" created "8 minutes ago". There is also a "Filter Tasks" input field.

You can see filter Task by Candidate user has on task which demo user can claim and complete.

## User Task with Candidate Groups:

## Task\_14qmcj3

General Forms Listeners Input/Output Extensions

### General

#### Id

Task\_14qmcj3 x

#### Name

User Task //

### Details

#### Assignee

#### Candidate Users

#### Candidate Groups

accounting x

#### Due Date

The due date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

#### Follow Up Date

The follow up date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

## Deploy the process

## Start the process

Start process

Search by process name.

 Click on the process to start.

Process User Task Candidate Grroup Demo

Process User Task Candidate User Demo

[Close](#)

## My Tasks

### My Group Tasks (1)

Accounting

John's Tasks

Mary's Tasks



Filter Tasks

1

#### User Task

Process User Task Candidate Ggroup Demo

Created 4 minutes ago

50

## Task Detail

### User Task

Process User Task Candidate Ggroup Demo

Set follow-up date

Set due date

Accounting

Claim

Form

History

Diagram

Description

First name

Last name

Save

Complete

User task is assigned to accounting group.

User can claim and complete the task

**User Task Due Date:**

Set Due Date to 2019-12-22T12:00:00

**User Task Follow Up Date:**

Set Follow Up Date to 2019-11-22T11:00:00

**Task\_14qmcj3**

General   Forms   Listeners   Input/Output   Extensions

**General**

**Id**  
Task\_14qmcj3

**Name**  
User Task

---

**Details**

**Assignee**  
[empty field]

**Candidate Users**  
[empty field]

**Candidate Groups**  
accounting

**Due Date**  
2019-12-22T12:00:00

The due date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

**Follow Up Date**  
2019-11-22T11:00:00

The follow up date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

**Deploy the process and Start process**

# User Task

Process User Task Candidate Grouop Demo [🔗](#)

📅 in 4 days ×

🔔 in a month ×

🏢 Accounting

👤 Claim

Form History Diagram Description

First name

Last name

Save

Complete

## Deploy User Task Form Field Validation

Validation can be used for specifying frontend and backend validation of form fields. Camunda BPM provides a set of built-in form field validators and an extension point for plugging in custom validators.

The following built-in validators are supported out of the box:

Validator	Explanation
required	Applicable to all types. Validates that a value is provided for the form field. Rejects 'null' values and empty strings.  eg. <camunda:constraint name="required" />
minlength	Applicable to string fields. Validates minlength of text content. Accepts 'null' values.  eg. <camunda:constraint name="minlength" config="4" />

maxlength	Applicable to string fields. Validates maxlenlength of text content. Accepts 'null' values.  eg. <camunda:constraint name="maxlength" config="25" />
min	Applicable to numeric fields. Validates the min value of a number. Accepts 'null' values.  eg. <camunda:constraint name="min" config="1000" />
max	Applicable to numeric fields. Validates the max value of a number. Accepts 'null' values.  eg. <camunda:constraint name="max" config="10000" />
readonly	Applicable to all type. Makes sure no input is submitted for given form field.  eg. <camunda:constraint name="readonly" />