Interreg
EUROPEAN UNION
ADRION ADRIATIC-IONIAN
European Regional Development Fund - Instrument for Pre-Accession II Fund

TRANSCPEARLYWARNING

TransCPEarlyWarning Civil Protection Early Warning Platform

TransCPEW Business Process Development MANUAL

"TRANSCPEARLYWARNING": Establishment of "TRANSnational Civil Protection EARLY WARNING System" to improve the resilience of Adrion territories to naturaland man-made risks.

JUNE 2022

Disclaimer

This document has been produced with the financial assistance of the European Union. The content of the document is the sole responsibility of ADRION TransCPEarlyWarning partnership under the coordination of partner Industrial Systems Institute for the compilation of the specific document and can under no circumstances be regarded as reflecting the position of the European Union and/or ADRION programme authorities.

TRANSCPEARLYWARNING: Establishment of "TRANSnational Civil Protection EARLY WARNING System" to improve the resilience of Adrion territories to natural and man-made risks (ADRION 979)

Programme Priority 2. Sustainable Region

Specific Objective: Enhance the capacity in transnationally tackling environmental
vulnerability, fragmentation, and the safeguarding of ecosystem services in the Adriatic-Ionian area
WP T2 – Civil Protection & Early Warning Platform linked to the EU Civil Protection Mechanisms
Activity T2.1 – Development of Civil Protection Early Warning Platform
Deliverable T2.1.2 – Civil Protection Early Warning Platform with semantics interface

Responsible Author: PP3 - Athanasios Kalogeras, Christos Anagnostopoulos, Agorakis Bombotas, Georgios Mylonas, Christos Alexakos, Kyriakos Stefanidis, Georgios Kalogeras, Georgios Raptis, Stella Markantonatou, Miranda Dandoulaki, Georgios Lefteriotis

Editors: External expert Dynamic Vision - Ioannis Mardikis, Natalia Tsami

Project Coordinator: ***LP – Regione Molise***

INTERREG ADRION
Programme

2014-2020

Contents

List of Figures

❖ Executive Summary

This manual elaborates on the model of the EUSAIR Civil Protection Early Warning System, focusing on the operation of process models which provide the necessary concepts and semantics for the envisaged Civil Protection Early Warning System Platform.

❖ Introduction

The **TransCPEW platform** aims to unify and automate the various Civil Protection (CP) processes regarding the prevention of natural and man-made disasters. **It serves the purpose of offering a focal point of reference for the Civil Protection stakeholders in ADRION territories** enabling the integration of different information sources and systems and will make it possible for CP stakeholders to perform the relevant experimentation through pilot implementations.

A Civil Protection Early Warning System essentially comprises a collection of processes in which information is exchanged between several actors in a predefined and organized fashion. The flow of information is time-dependent in most cases and involves several stakeholders with different responsibilities. Each stakeholder has a special reaction, dependent on the type of message he receives. It is obvious at this point that an EWS is a complex system that presents several challenges when trying to understand and integrate it into a web platform.

An extra degree of complexity is added when taking into account not just one but several different Early Warning Systems from the different partner countries. These countries have different organizational structures and follow different processes when dealing with the early warning part of a natural disaster. The number of involved actors can also vary greatly and the information flow does not always follow the same path.

Taking all the above into consideration, it is evident that the system has to be modelled in order to create a concrete base capturing the general principles behind the Early Warning Systems of the involved countries. An ontological model fits this purpose precisely, offering a formal specification of the terms in a specific domain and the relationships between them. The main goal of the model at hand is to support the sharing of a common understanding of the structure and flow of information of the EWS.

- The model attempts to answer to questions such as:
- What messages are created and exchanged in the system?
- Which stakeholders are involved in various steps of the process?
- What levels of administration the various stakeholders belong to?
- How does the information flow from level to level and to which direction?
- Are there any time related constraints in the process workflow?

The model answers these questions in a formal manner, easily reproducible and exchangeable between system agents and at the same it provides the basis for the implementation of a unified platform that can host the Early Warning Systems of all the partners. In order to achieve this, the system also answers another set of questions, such as:

- What are the differences among the actors involved in EWS of each country?
- How flow of information differs among countries?
- Are countries utilizing the same levels of administrative stakeholders in the process execution and information flow?

The TransCPEW model is a result of analysis of the different concepts implemented by the Civil Protection Early Warning processes and systems in the different countries. This model has been designed to be applicable in the area as a whole.

The current document, "TransCPEW Business Process Development MANUAL", addresses experts authorized for developing a Business Process.

1. TransCPEarlyWarning business process engine

1.1. Overview

In this chapter we are going to describe the part of the TransCPEarlyWarning platform which is responsible for the modeling, the implementation, and the execution of the Civil Protection business processes. An overview of the architecture of this part is presented in ***Figure 4***.

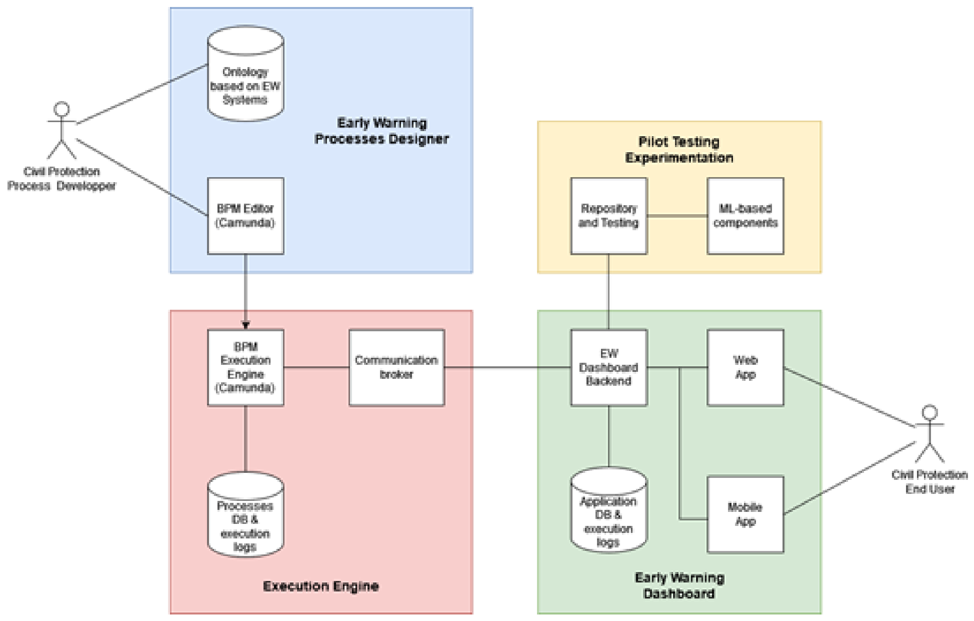

Ontology based on EW Systems
Early Warning Processes Designer
Civil Protection Process Developper
BPM Editor (Camunda)
Pilot Testing Experimentation
Repository and Testing
ML-based components
BPM Execution Engine (Camunda)
Communication broker
EW Dashboard Backend
Web App
Civil Protection End User
Processes DB & execution logs
Application DB & execution logs
Mobile App
Execution Engine
Early Warning Dashboard

Figure 1: Part of the TransCPEarlyWarning platform under examination

The user who is going to interact with this part is the Civil Protection Process Developer and not a simple TransCPEarlyWarning platform user. This means that the business process development tools and workflows, which are described, are targeted to individuals who are familiar to some extent with assembling business processes using the BPMN 2.0 specification and working with java-based technologies. As depicted in ***Figure 2***, the business process workflow can be split into two parts. The first part is the modeling of the Civil Protection Operation Plan using the BPMN 2.0 notation and the second part is the development of the executable process that can be run in the business process engine.

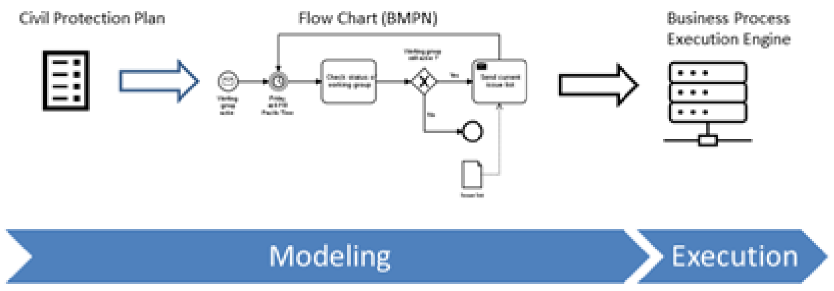


Figure 2: Business Process Modeling Workflow

For implementing the aforementioned tasks, we have chosen **Camunda**[1]**,** which is an open-source java-based platform that supports Business Process Model and Notation (BPMN 2.0), Case Management and Model Notation and Decision (CMMN 1.1) and Model Notation (DMN 1.3) specifications for creating and managing workflows.

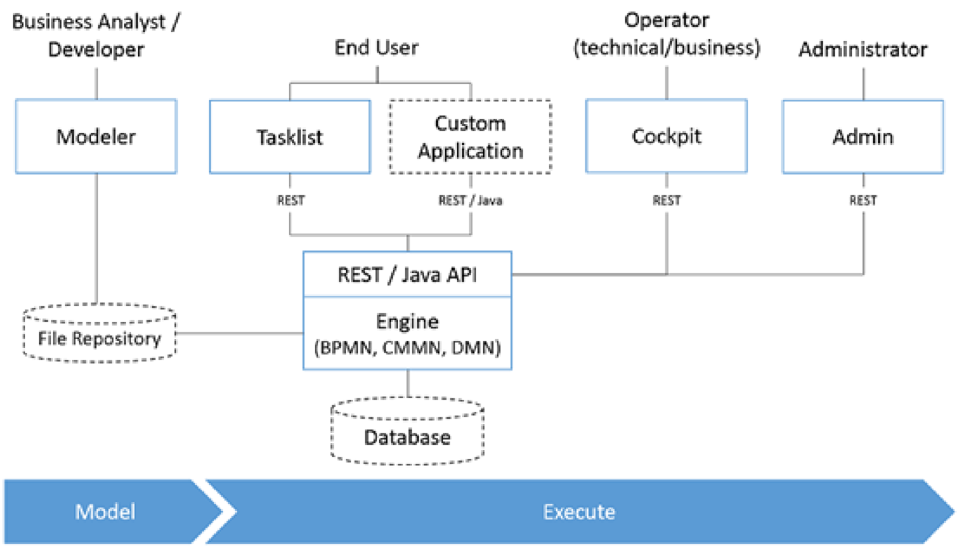


Figure 3: Basic components and indicative user roles of the Camunda platform (source: https://docs.camunda.org/manual/7.16/introduction)

As illustrated in ***Figure 3,*** Camunda components can be categorized into three categories:

- **Camunda Modeler**
- **Web Apps**
- **Process Engine**

[1] https://camunda.com/

1.2. Camunda modeler

The business process engine needs executable processes already designed and configured. Camunda modeler is a modelling tool for developing such models, which are compliant with Business Process Model and Notation (BPMN) 2.0, Case Management and Model Notation (CMMN) 1.1 and Decision and Model Notation (DMN) 1.3 specifications.

Camunda modeler is officially supported on the following operating systems:

- Windows 7/10
- Mac OS X 10.11 and later
- Ubuntu LTS (latest)

1.2.1. Download and installation

The tool can be downloaded from this link https://camunda.com/download/modeler/. After the installation the user must unzip the downloaded content in their folder of preference. There is no need for any additional installation process. After unzipping, the appropriate file depending on the hosting OS (Camunda Modeler.exe for Windows, Camunda Modeler.app for Mac and camunda-modeler for Linux based systems) must be executed.

1.2.2. Main Window

The main window of the Modeler is displayed in ***Figure 7*.** On top there is the **basic menu** that includes all the basic functionalities like creating a new diagram, opening a new one etc.

On the left there is the **tools panel** that includes all the basic types of BPMN components. You can click and drag and drop each tile on the main drawing canvas area. If you hover the mouse pointer over each tile an informative tooltip will appear.

On the bottom there is a ribbon which includes the following buttons:

- Button for toggling between XML view and normal view
- Button for deploying the current BPMN diagram to Camunda Platform
- Button for initiating the process described by the current diagram
- Button for toggling the Logging window

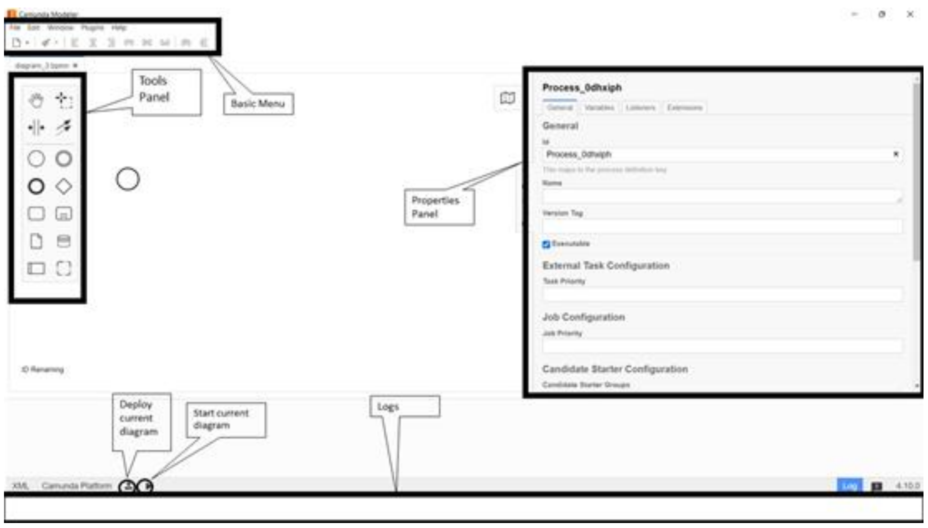


Figure 4: Camunda Modeler window

1.2.3. Using the main menu

A user can create a new BPMN diagram by selecting New File>BPMN Diagram (Camunda Platform) from the File menu. Respectively an already created diagram can be opened.

1.2.4. Tools palette

The palette on the left side of the modeler contains a set of tiles which can be dragged and dropped onto the main diagram canvas. The tiles include main manipulation tools and BPMN elements.

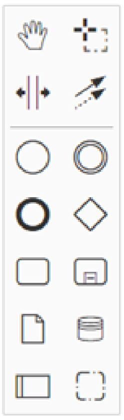
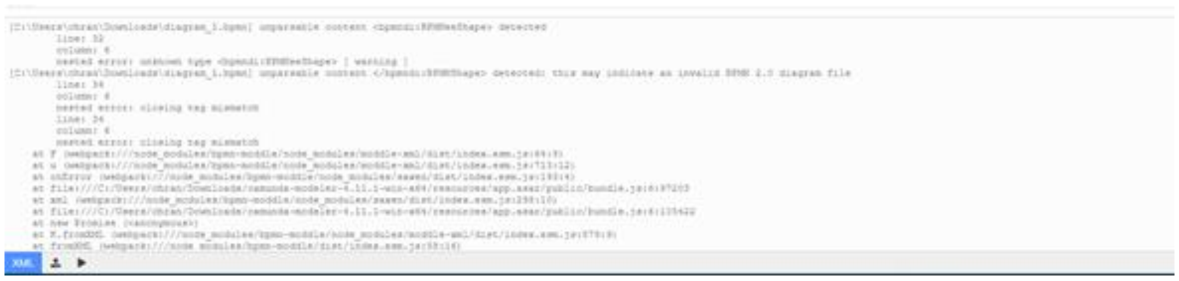


Figure 5: Tools Palette

1.2.5. Logging window

This window is at the bottom of the Camunda modeler and provides valuable information regarding error messages and deployment information.

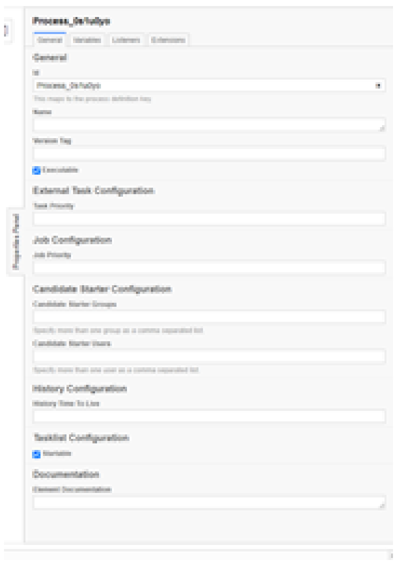


```
[C:\Users\chran\Downloads\diagram_1.bpmn] unparsable content <bpmndi:BPMNEdgeShape> detected
        line: 32
        column: 6
        nested error: unknown type <bpmndi:BPMNEdgeShape> [ warning ]
[C:\Users\chran\Downloads\diagram_1.bpmn] unparsable content </bpmndi:BPMNShape> detected; this may indicate an invalid BPMN 2.0 diagram file
        line: 34
        column: 6
        nested error: closing tag mismatch
        line: 34
        column: 6
        nested error: closing tag mismatch
    at F (webpack:///node_modules/bpmn-moddle/node_modules/moddle-xml/dist/index.esm.js:69:7)
    at u (webpack:///node_modules/bpmn-moddle/node_modules/moddle-xml/dist/index.esm.js:713:12)
    at onError (webpack:///node_modules/bpmn-moddle/node_modules/saxen/dist/index.esm.js:193:4)
    at file:///C:/Users/chran/Downloads/camunda-modeler-4.11.1-win-x64/resources/app.asar/public/bundle.js:6:97203
    at xml (webpack:///node_modules/bpmn-moddle/node_modules/saxen/dist/index.esm.js:298:10)
    at file:///C:/Users/chran/Downloads/camunda-modeler-4.11.1-win-x64/resources/app.asar/public/bundle.js:6:135422
    at new Promise (<anonymous>)
    at F.fromXML (webpack:///node_modules/bpmn-moddle/node_modules/moddle-xml/dist/index.esm.js:879:8)
    at fromXML (webpack:///node_modules/bpmn-moddle/dist/index.esm.js:58:16)
```

Figure 6: Error message

1.2.6. Properties Panel

When you click on an item in the diagram canvas, the properties panel contains all the configurable information concerning the specific element.



Process_0s1u0yo

General | Variables | Listeners | Extensions

General

Id

Process_0s1u0yo

This maps to the process definition key.

Name

Version Tag

☑ Executable

External Task Configuration

Task Priority

Job Configuration

Job Priority

Candidate Starter Configuration

Candidate Starter Groups

Specify more than one group as a comma separated list.

Candidate Starter Users

Specify more than one user as a comma separated list.

History Configuration

History Time To Live

Tasklist Configuration

☑ Startable

Documentation

Element Documentation

Figure 7: Properties Panel

1.3. Web Apps

Camunda provides a set of web applications for interacting with the process engine either manually or programmatically. These applications are:

- Camunda Tasklist for monitoring tasks and initiating processes.
- Camunda Cockpit for monitoring workflows and deployments
- Camunda Admin for managing users.
- REST API for accessing the process engine remotely.

The communication between the TransCPEarlyWarning fron-end and the TransCPEarlyWarning business process engine is executed via the REST API.

1.4. Process Engine

The process engine is a Java library which can execute BPMN 2.0 processes. For storage it utilizes a relational database and it uses a lightweight POJO core. The MyBatis mapping framework provides the ORM mapping.

A **process definition** is the description of a business process, and the engine uses BPMN 2.0 as the modeling language. The user deploys the processes to the engine in BPMN 2.0 XML format, which is parsed to a process definition graph structure. Each process definition bears a **key which** identifies her unambiguously. The instantiation of a process definition creates a process instance which is the executable of the business process. From a process definition multiple process instances can be created. The process instances can be started by an external trigger (e.g., a time event) or from a user action in the TransCPEarlyWarning front-end, using the REST API. Each process instance contains a set of process variables which are also accessible. Different process instances communicate with each other via message and signal events.

1.5. Business Process Development

The development and deployment of a process definition can be done either by using only the Camunda Modeler or by developing a Java Process Application using Eclipse IDE. In the latter case you can fully exploit the capabilities of Java but the implementation effort is larger.

The development of the Business processes is described in full detail in the online Camunda Platform Manual. However, in the following two chapters we present the most important steps associated with this procedure.

1.5.1. Using Camunda Modeler

Create a new BPMN diagram

For creating a new BPMN 2.0 diagram Select New File>BPMN Diagram (Camunda Platform) from the File menu.

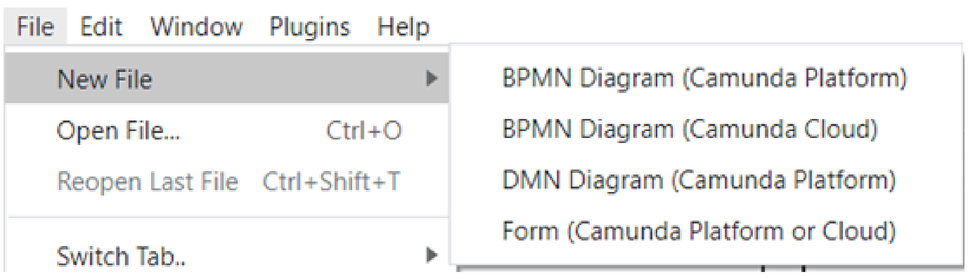

File
Edit
Window
Plugins
Help
New File
Open File... Ctrl+O
Reopen Last File Ctrl+Shift+T
Switch Tab..
BPMN Diagram (Camunda Platform)
BPMN Diagram (Camunda Cloud)
DMN Diagram (Camunda Platform)
Form (Camunda Platform or Cloud)

Figure 8: New BPMN Diagramm Creation

Create a Simple Process

1. Drag and drop a StartEvent component from the tools palette.
2. Click on the palette and select the Append Task option

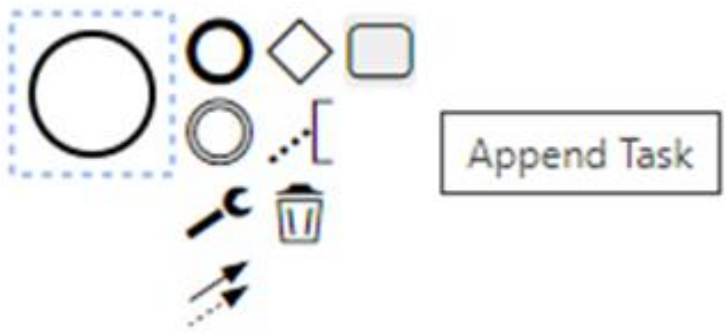


Figure 9: Simple Process Creation

3. Similarly, or by using the tools palette create the desirable BPMN process.
4. By clicking on each Task and selecting the wrench tool you can change the type of the Task.

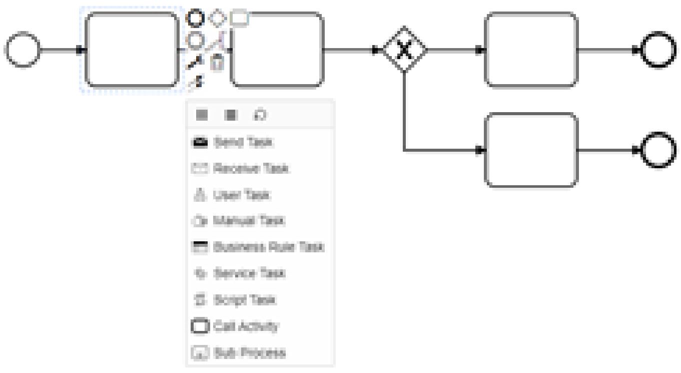
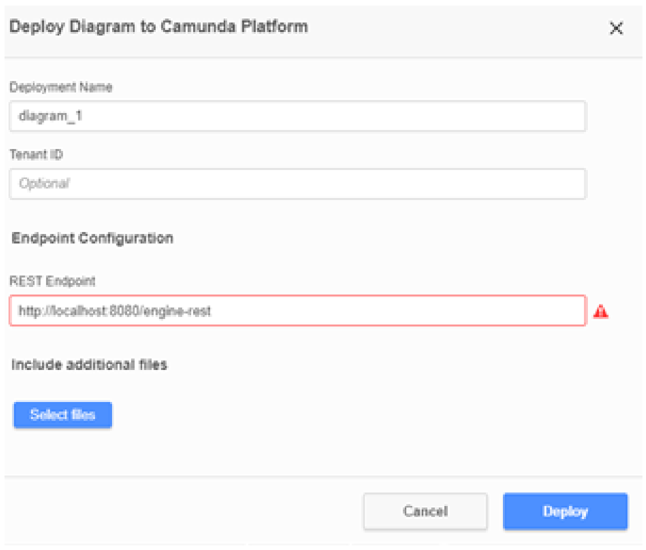


Figure 10: Types of Tasks

5. You can view and edit the properties of the currently selected component in the properties panel.
6. When you are ready save the diagram using the main menu.

Deployment

1. Click on the Deployment button on the Camunda modeler
2. On the new window fille the appropriate fields. Include any additional files if needed.


Deploy Diagram to Camunda Platform

Deployment Name
diagram_1

Tenant ID
Optional

Endpoint Configuration

REST Endpoint
http://localhost:8080/engine-rest

Include additional files

Figure 11: Deployment

3. If everything is fine a success message will be displayed in the modeler.

1.5.2. Java Process Application

Since you have already installed the Camunda Platform and the Camunda Modeler the following tools must be installed:

- Java JDK 1.8+
- Apache Maven (embedded Maven in Eclipse may be used)
- A web browser
- Eclipse IDE

Maven is an open-source tool developed by Apache Group, used for building, managing, and deploying Java-based projects. It is written in Java, and it is based on the Project Object Model (POM), which is an XML file containing all the information regarding project and configuration details. The relevant pom.xml file is in the project's root folder.

1.5.2.1. Installing Maven Project Templates (Archetypes)

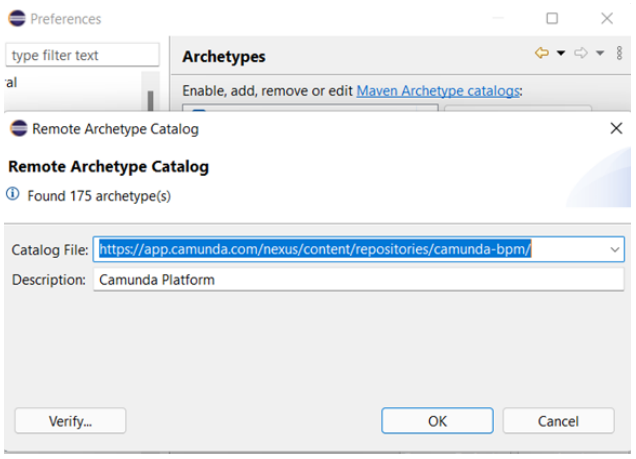
Maven project templates or Archetypes are used for the rapid development of production-ready process applications using the Camunda Platform. They are used for the generation of projects for different use cases. Currently the following archetypes are available in Camunda's Maven repository.

- **Camunda Cockpit Plugin**: A plugin for Camunda Cockpit.
- **Process Application (EJB, WAR)**: A process application that uses a shared Camunda Platform engine in a Java EE Container.
- **Process Application (Servlet, WAR): A** process application that uses a shared Camunda Platform engine in a Servlet Container.
- **Camunda Spring Boot Application**: An application that uses the Camunda Spring Boot Starter
- **Camunda Spring Boot Application with Demo Users**: Same as the previous archetype, but in addition create demo users.
- **Process Engine Plugin:** An example of a process engine plugin.

For installing the Camunda Maven Archetypes to Eclipse IDE, after starting the IDE, go to *Window -> Preferences -> Maven -> Archetypes -> Add Remote Catalog* and enter the following:

- **Catalog File**: https://app.camunda.com/nexus/content/repositories/camunda-bpm/
- **Description**: Camunda Platform

Click *Verify* to finish with the installations of the templates. Now the Archetypes are ready to be used.



Remote Archetype Catalog
Found 175 archetype(s)

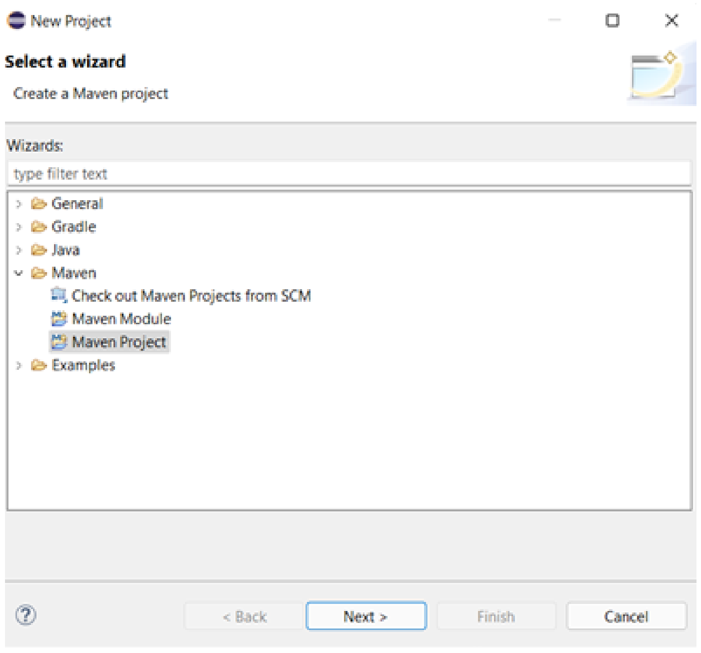
Catalog File: https://app.camunda.com/nexus/content/repositories/camunda-bpm/
Description: Camunda Platform

Figure 12: Installing Maven

1.5.2.2. Setting up a Maven Project

For setting up a new project follow the steps described below

1. Create a new Maven project using the Archetypes installed in the previous chapter going to *File -> New -> Project,* select *Maven->Maven Project* and click *Next.*


New Project

Select a wizard

Create a Maven project

Wizards:

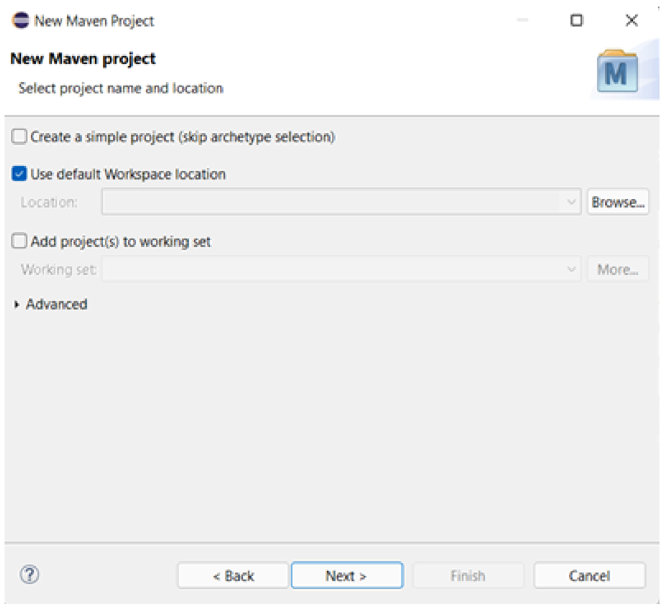
type filter text

- General
- Gradle
- Java
- Maven
  - Check out Maven Projects from SCM
  - Maven Module
  - Maven Project
- Examples

< Back | Next > | Finish | Cancel

Figure 13: : Set up a Maven Project -> Step 1

2. In the following screen use the default settings or change the location of the project. When you have finished press *Next.*


New Maven Project

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

☑ Use default Workspace location

Location: Browse...

☐ Add project(s) to working set

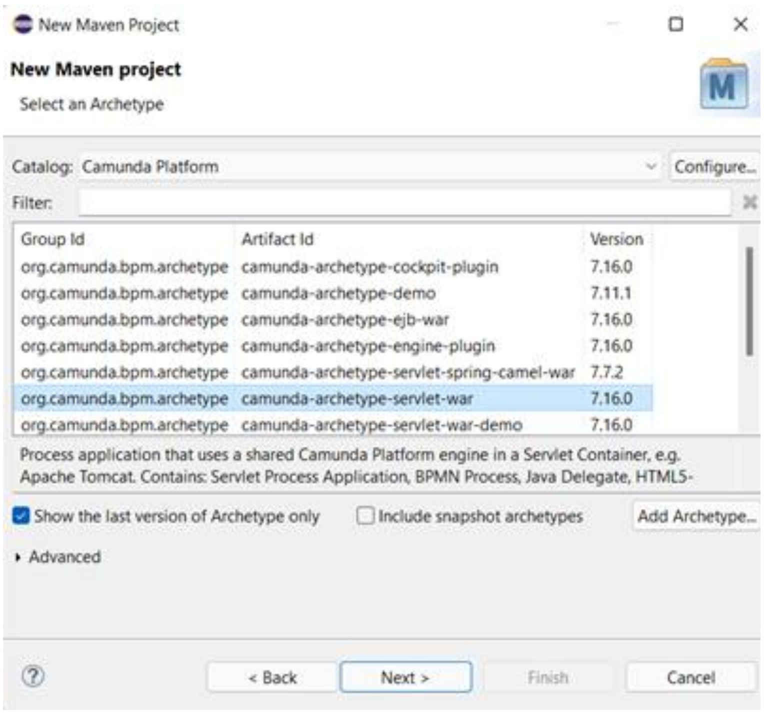
Working set: More...

Advanced

< Back | Next > | Finish | Cancel

Figure 14: Set up a Maven Project -> Step 2

3. Select the archetype with Artifact ID “camunda-archetype-servlet-war” and choose appropriate version of the Camunda Platform. In our case it is 7.16. Press *Next* when you're done.


New Maven Project

New Maven project

Select an Archetype

Catalog: Camunda Platform

Filter:

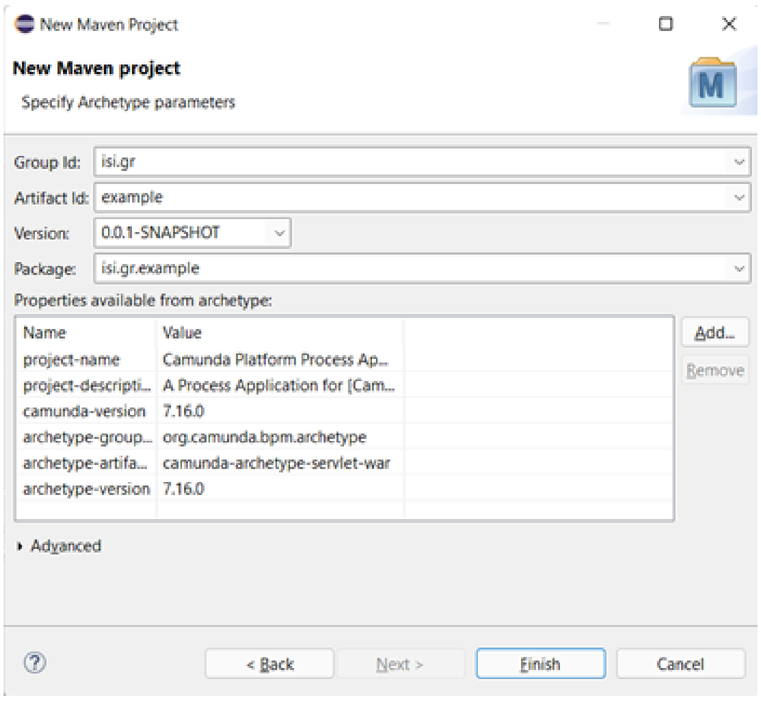
Group Id	Artifact Id	Version
org.camunda.bpm.archetype	camunda-archetype-cockpit-plugin	7.16.0
org.camunda.bpm.archetype	camunda-archetype-demo	7.11.1
org.camunda.bpm.archetype	camunda-archetype-ejb-war	7.16.0
org.camunda.bpm.archetype	camunda-archetype-engine-plugin	7.16.0
org.camunda.bpm.archetype	camunda-archetype-servlet-spring-camel-war	7.7.2
org.camunda.bpm.archetype	camunda-archetype-servlet-war	7.16.0
org.camunda.bpm.archetype	camunda-archetype-servlet-war-demo	7.16.0

Process application that uses a shared Camunda Platform engine in a Servlet Container, e.g. Apache Tomcat. Contains: Servlet Process Application, BPMN Process, Java Delegate, HTML5-

[✓] Show the last version of Archetype only [ ] Include snapshot archetypes

Figure 15: Set up a Maven Project -> Step 3

4. Fill the necessary fields in the following form and click the *Finish* button.


New Maven Project

New Maven project

Specify Archetype parameters

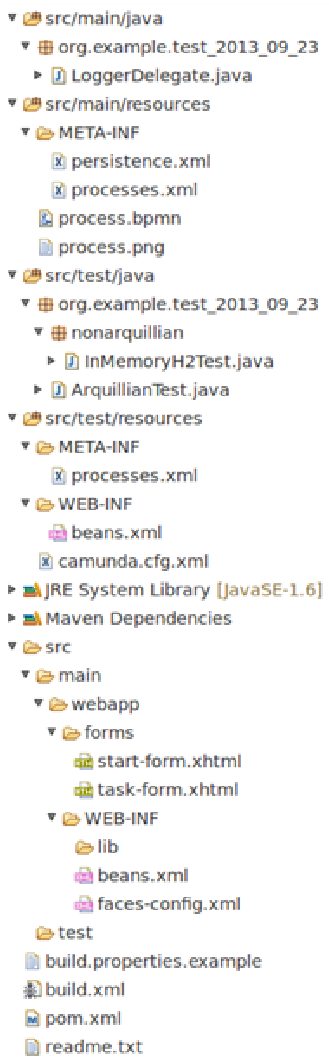
Group Id:	isi.gr
Artifact Id:	example
Version:	0.0.1-SNAPSHOT
Package:	isi.gr.example

Properties available from archetype:

Name	Value
project-name	Camunda Platform Process Ap...
project-descripti...	A Process Application for [Cam...
camunda-version	7.16.0
archetype-group...	org.camunda.bpm.archetype
archetype-artifa...	camunda-archetype-servlet-war
archetype-version	7.16.0

Figure 16: Set up a Maven Project -> Step 4

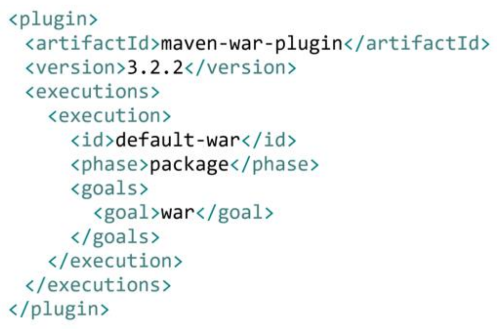
5. The structure of the project is the following



```
▾ src/main/java
  ▾ org.example.test_2013_09_23
    ▸ LoggerDelegate.java
▾ src/main/resources
  ▾ META-INF
      persistence.xml
      processes.xml
    process.bpmn
    process.png
▾ src/test/java
  ▾ org.example.test_2013_09_23
    ▾ nonarquillian
      ▸ InMemoryH2Test.java
    ▸ ArquillianTest.java
▾ src/test/resources
  ▾ META-INF
      processes.xml
  ▾ WEB-INF
      beans.xml
    camunda.cfg.xml
▸ JRE System Library [JavaSE-1.6]
▸ Maven Dependencies
▾ src
  ▾ main
    ▾ webapp
      ▾ forms
          start-form.xhtml
          task-form.xhtml
      ▾ WEB-INF
          lib
          beans.xml
          faces-config.xml
    test
  build.properties.example
  build.xml
  pom.xml
  readme.txt
```

Figure 17: Set up a Maven Project -> Step 5

6. Finally, open the pom.xml file under the root project folder and add the following code under <plugins>.



```
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <version>3.2.2</version>
  <executions>
    <execution>
      <id>default-war</id>
      <phase>package</phase>
      <goals>
        <goal>war</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Figure 18: Set up a Maven Project -> Step 6

1.5.2.3. Model a process

To start modeling a process open the *process.bpmn* file under */src/main/resources* using Camunda Modeler and following the steps as described in the previous chapter. When the process is ready save the file.

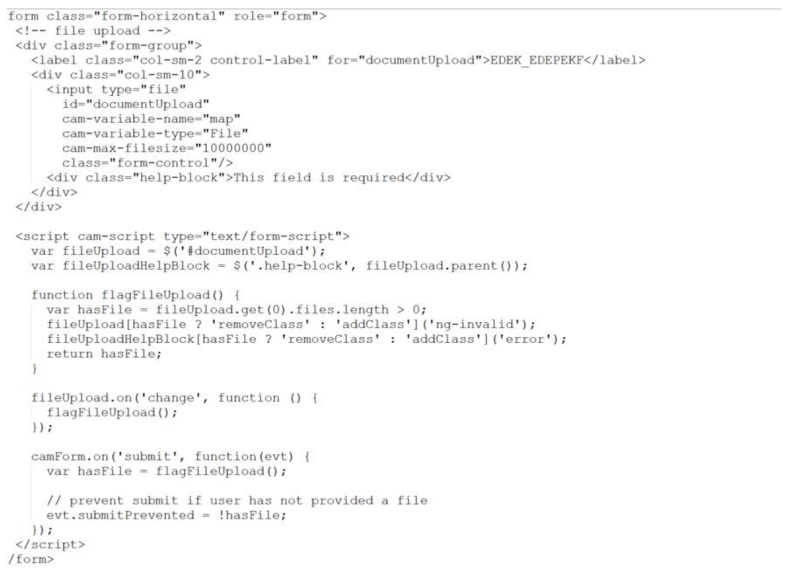
1.5.2.4. Building and deployment

For the application using Maven select pom.xml file in the Package explorer of the Eclipse IDE and from the context menu select *Run As / Maven install.* This will result in generating a WAR file under the target folder of your project structure. The bpmn file is included in this file too.

In order to deploy the application to the Business process engine you have to copy the generate war file to the *$CAMUNDA_HOME/server/apache-tomcat/webapp* folder which is located in the Apache Tomcat server hosting the Camunda Platform.

1.5.2.5. Add a form

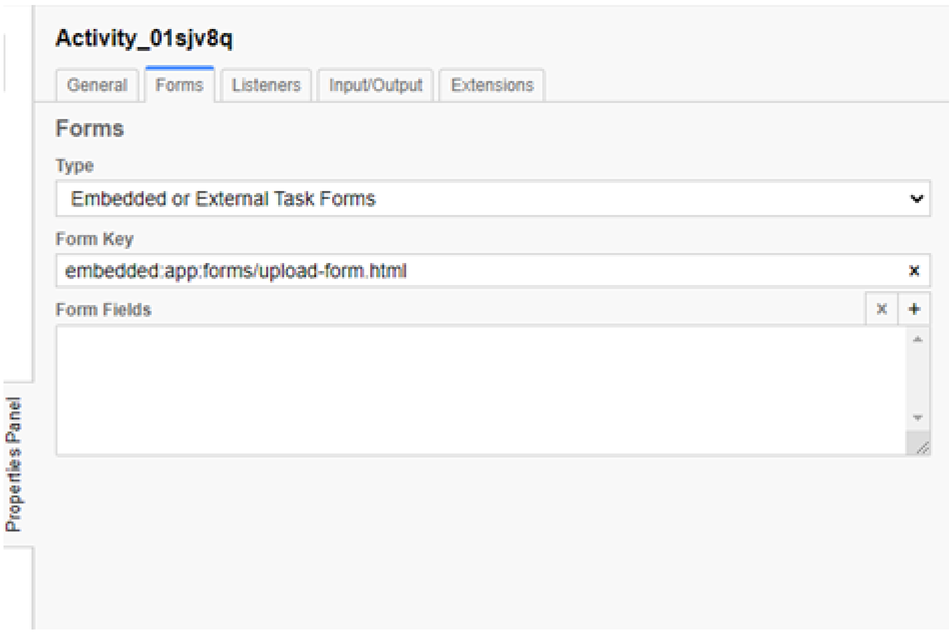
In Eclipse and under the Package Explorer window create a new folder named *src/main/webapp/forms*. Add and HTML file under this folder with the following content. The following code creates a form for uploading an object and assigning the uploaded file to the process variable "map". Save the file as upload-form.html



```
form class="form-horizontal" role="form">
 <!-- file upload -->
 <div class="form-group">
   <label class="col-sm-2 control-label" for="documentUpload">EDEK_EDEPEKF</label>
   <div class="col-sm-10">
     <input type="file"
       id="documentUpload"
       cam-variable-name="map"
       cam-variable-type="File"
       cam-max-filesize="10000000"
       class="form-control"/>
     <div class="help-block">This field is required</div>
   </div>
 </div>

 <script cam-script type="text/form-script">
   var fileUpload = $('#documentUpload');
   var fileUploadHelpBlock = $('.help-block', fileUpload.parent());

   function flagFileUpload() {
     var hasFile = fileUpload.get(0).files.length > 0;
     fileUpload[hasFile ? 'removeClass' : 'addClass']('ng-invalid');
     fileUploadHelpBlock[hasFile ? 'removeClass' : 'addClass']('error');
     return hasFile;
   }

   fileUpload.on('change', function () {
     flagFileUpload();
   });

   camForm.on('submit', function(evt) {
     var hasFile = flagFileUpload();

     // prevent submit if user has not provided a file
     evt.submitPrevented = !hasFile;
   });
 </script>
/form>
```

Figure 19: Add the HTML file

For assigning the form to a User Task, open the process.bpmn file using the Modeler and select the User Task that you want to assign this form to. On the properties panel select the Forms Tab and under Type select Embedded or External Task Forms and under Form Key type: *embedded:app:forms/upload-form.html*.



Activity_01sjv8q

Forms

Type: Embedded or External Task Forms

Form Key: embedded:app:forms/upload-form.html

Form Fields

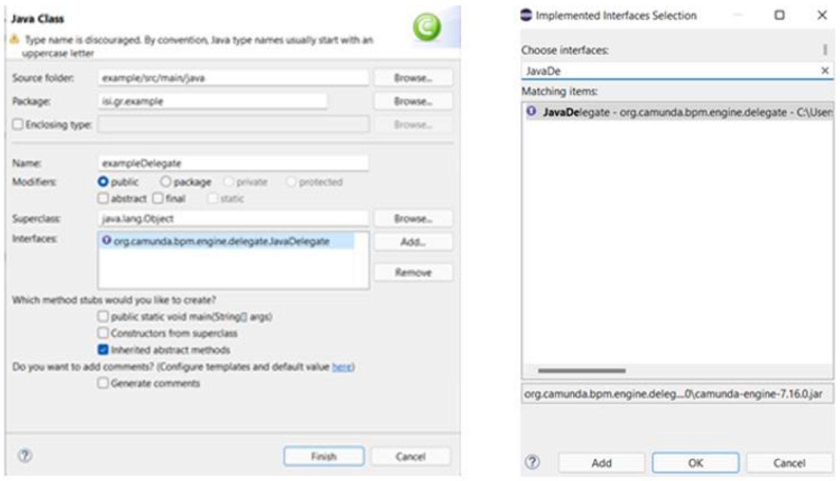
Figure 20: Add a form

When you are ready re-build and re-deploy the project.

1.5.2.6. Add a Java delegate

In this part we will demonstrate how to invoke a Java class from a BPMN process.

1. Open the process.bpmn using Camunda Modeler and create a Service Task either by dragging and dropping one from the Tools Palette or by changing an already deployed Task by using the wrench tool.
2. Select the Service Task and on the Properties Panel Under the Details section select "Java Class" under Implementation.
3. In Eclipse under you src folder create a new Java Class by right clicking and selecting New -> Class. In the newly created windows click on the button *Add.*
4. Select the JavaDelegate from the list of all the interfaces and click *OK* to close the window. Click Finish on the remaining window to finish the process.



Java Class

Type name is discouraged. By convention, Java type names usually start with an uppercase letter

Source folder: example/src/main/java

Package: isi.gr.example

Enclosing type:

Name: exampleDelegate

Modifiers: [✓ public] package private protected

abstract final static

Superclass: java.lang.Object

Interfaces: org.camunda.bpm.engine.delegate.JavaDelegate

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

[✓ Inherited abstract methods]

Do you want to add comments? (Configure templates and default value here)

Generate comments

Implemented Interfaces Selection

Choose interfaces: JavaDe

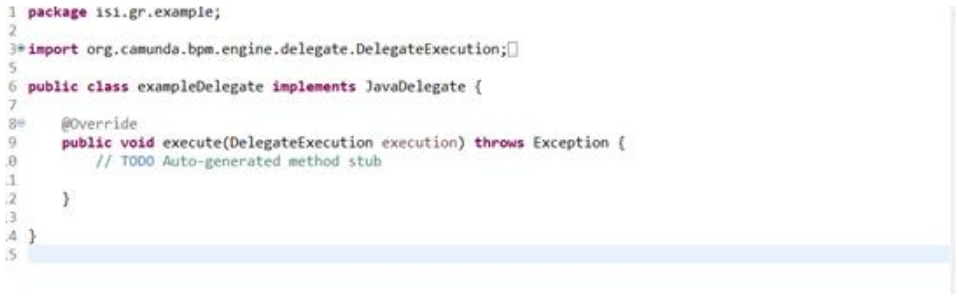
Matching items:

JavaDelegate - org.camunda.bpm.engine.delegate - C:\User

org.camunda.bpm.engine.deleg...0\camunda-engine-7.16.0.jar

Figure 21: Add a Java delegate -> Step 3,4

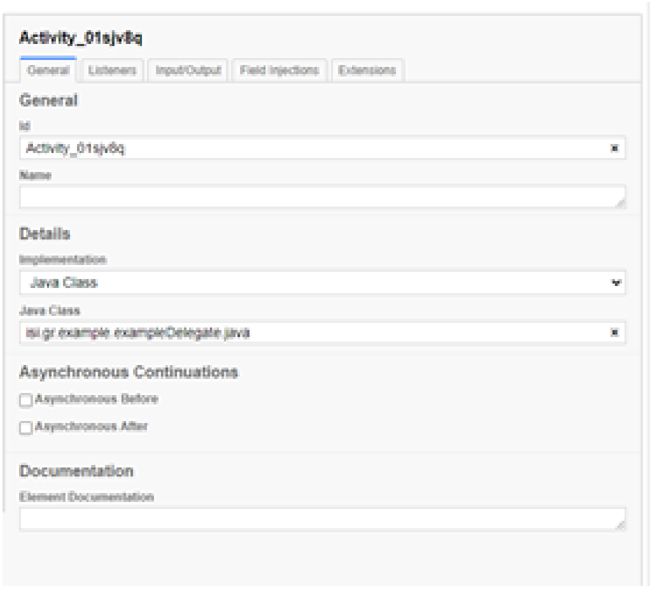
5. Open the newly created exampleDelegate.java file in Eclipse and implement the logic inside the execute function. This function is called when the Service Task is activated.



```
package isi.gr.example;

import org.camunda.bpm.engine.delegate.DelegateExecution;

public class exampleDelegate implements JavaDelegate {

    @Override
    public void execute(DelegateExecution execution) throws Exception {
        // TODO Auto-generated method stub

    }

}
```

Figure 22: Add a Java delegate -> Step 5

6. Right click the parent folder of the exampleDelegate.java and select Copy Qualified Name
7. Go to the modeler, select the Service Task and in the properties Panel under the General Tab, paste the Qualified name under the Label Java Class. At the end add ".<name of the java file>".



Activity_01sjv8q

General | Listeners | Input/Output | Field Injections | Extensions

General

Id: Activity_01sjv8q

Name:

Details

Implementation: Java Class

Java Class: isi.gr.example.exampleDelegate.java

Asynchronous Continuations

☐ Asynchronous Before

☐ Asynchronous After

Documentation

Element Documentation:

Figure 23: Add a Java delegate -> Step 7

8. Save everything, build, and deploy as usual.

1.5.2.7. Add an additional resource

Here we will describe how to add an additional resource, like a json configuration file, to the Maven project.

1. Create a new json file called *example.json* under *src/main/resources*
2. Open file *process.xml* under src/main/resources/META-INF
3. Add <resource>example.json</resource
4. Build and deploy.