

# Flowable6.4 –

Flowable6.4 –

## Flowable

org.flowable.image.impl.DefaultProcessDiagramGenerator

1. BpmnModel
2. BpmnModel
3. Flowable Api

```
/**
 * @param type      (png jpg)
 * @param modelId
 * @param response HttpServletResponse
 */
@ResponseBody
@RequestMapping(value = "diagram/{type}/{modelId}")
public void getJpgDiagram(@PathVariable(value = "type") String type,
                          @PathVariable(value = "modelId") String
modelId,
                          HttpServletResponse response) {
    try {
        // modelId BpmnModel
        Model modelData = repositoryService.getModel(modelId);
        ExtBpmnJsonConverter jsonConverter = new ExtBpmnJsonConverter();
        byte[] modelEditorSource =
repositoryService.getModelEditorSource(modelData.getId());
        JsonNode editorNode = new ObjectMapper().readTree(modelEditorSource);
        BpmnModel bpmnModel = jsonConverter.convertToBpmnModel(editorNode);
        //
        DefaultProcessDiagramGenerator diagramGenerator = new
DefaultProcessDiagramGenerator();
        InputStream inputStream = diagramGenerator.generateDiagram(
            bpmnModel,
            type,
```

```

        Collections.emptyList(),
        Collections.emptyList(),
        " ",
        " ",
        " ",
        null,
        1.0,
        false);

    //
    IOUtils.copy(inputStream, response.getOutputStream());
    response.setHeader("Content-Disposition", "attachment; filename=" +
bpmnModel.getMainProcess().getId() + "." + type);
    response.flushBuffer();
} catch (Exception e) {
    e.printStackTrace();
}
}

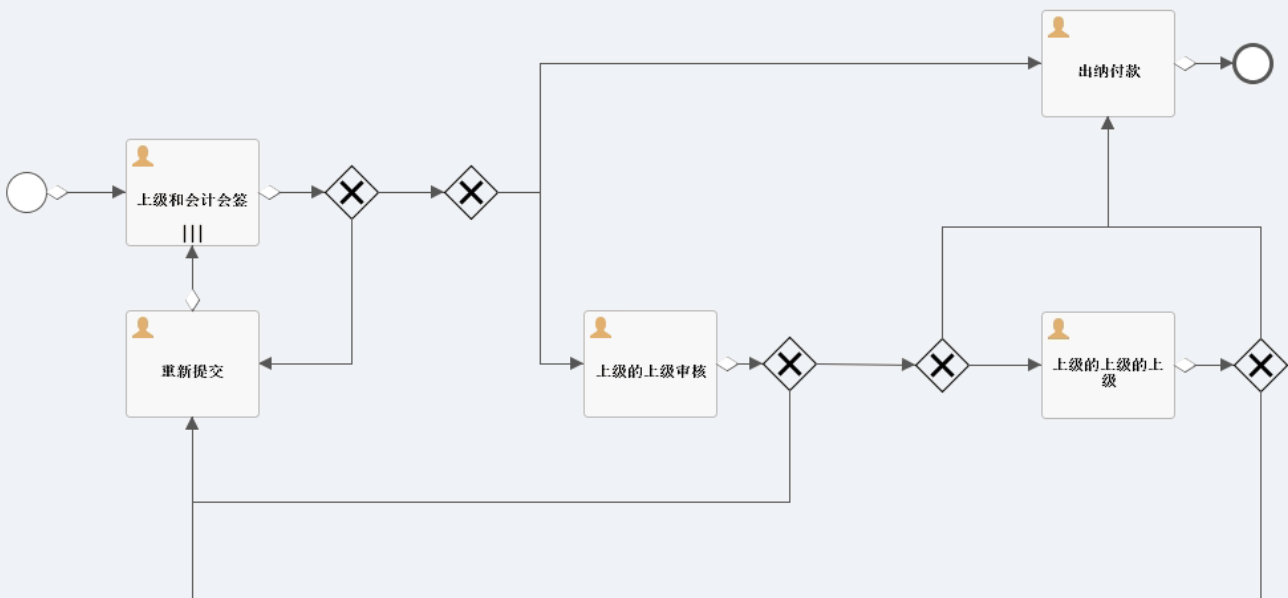
```

## Configuration

```

configuration.setActivityFontName(" ");
configuration.setLabelFontName(" ");

```



Sequence Flow

diagramGenerator.generateDiagram

emptyList

```

/**
 * @param type          (png jpg)
 * @param modelId
 * @param instId
 * @param response HttpServletResponse
 */
@ResponseBody
@RequestMapping( value = "activityDiagram/{type}/{modelId}/{instId}")
public void getJpgActivityDiagram(@PathVariable( value = "type") String
type,

                                @PathVariable( value = "modelId") String
modelId,

                                @PathVariable( value = "instId") String
instId,

                                HttpServletResponse response) {

    //
    List<HistoricActivityInstance> activityInstances =
historyService.createHistoricActivityInstanceQuery().processInstanceId( instId).orderByHistoricA
    List<String> activities = new ArrayList<>();
    List<String> flows = new ArrayList<>();
    for (HistoricActivityInstance activityInstance : activityInstances) {
        if ("sequenceFlow".equals(activityInstance.getActivityType())) {
            //
            flows.add( activityInstance.getActivityId());
        } else {
            //
            activities.add( activityInstance.getActivityId());
        }
    }

    try {
        // modelId BpmnModel
        Model modelData = repositoryService.getModel( modelId);
        ExtBpmnJsonConverter jsonConverter = new ExtBpmnJsonConverter();
        byte[] modelEditorSource =
repositoryService.getModelEditorSource( modelData.getId());
        JsonNode editorNode = new ObjectMapper().readTree( modelEditorSource);
        BpmnModel bpmnModel = jsonConverter.convertToBpmnModel( editorNode);
        //

```

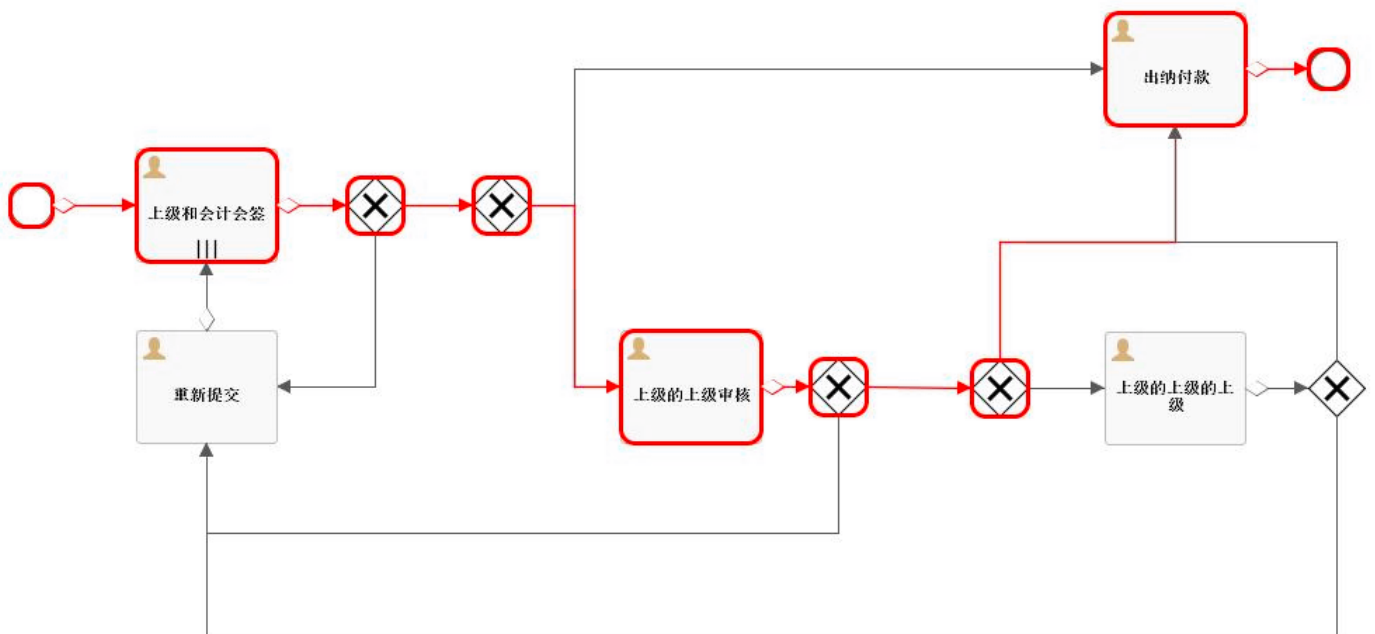
```

DefaultProcessDiagramGenerator diagramGenerator = new
DefaultProcessDiagramGenerator();

    InputStream inputStream = diagramGenerator.generateDiagram(
        bpmnModel,
        type,
        activities,
        flows,
        " ",
        " ",
        " ",
        null,
        1.0,
        false);

    //
    IOUtils.copy(inputStream, response.getOutputStream());
    response.setHeader("Content-Disposition", "attachment; filename=" +
bpmnModel.getMainProcess().getId() + "." + type);
    response.flushBuffer();
} catch (Exception e) {
    e.printStackTrace();
}
}

```



Revision #1

Created 30 July 2020 14:54:45 by

Updated 30 July 2020 14:56:35 by