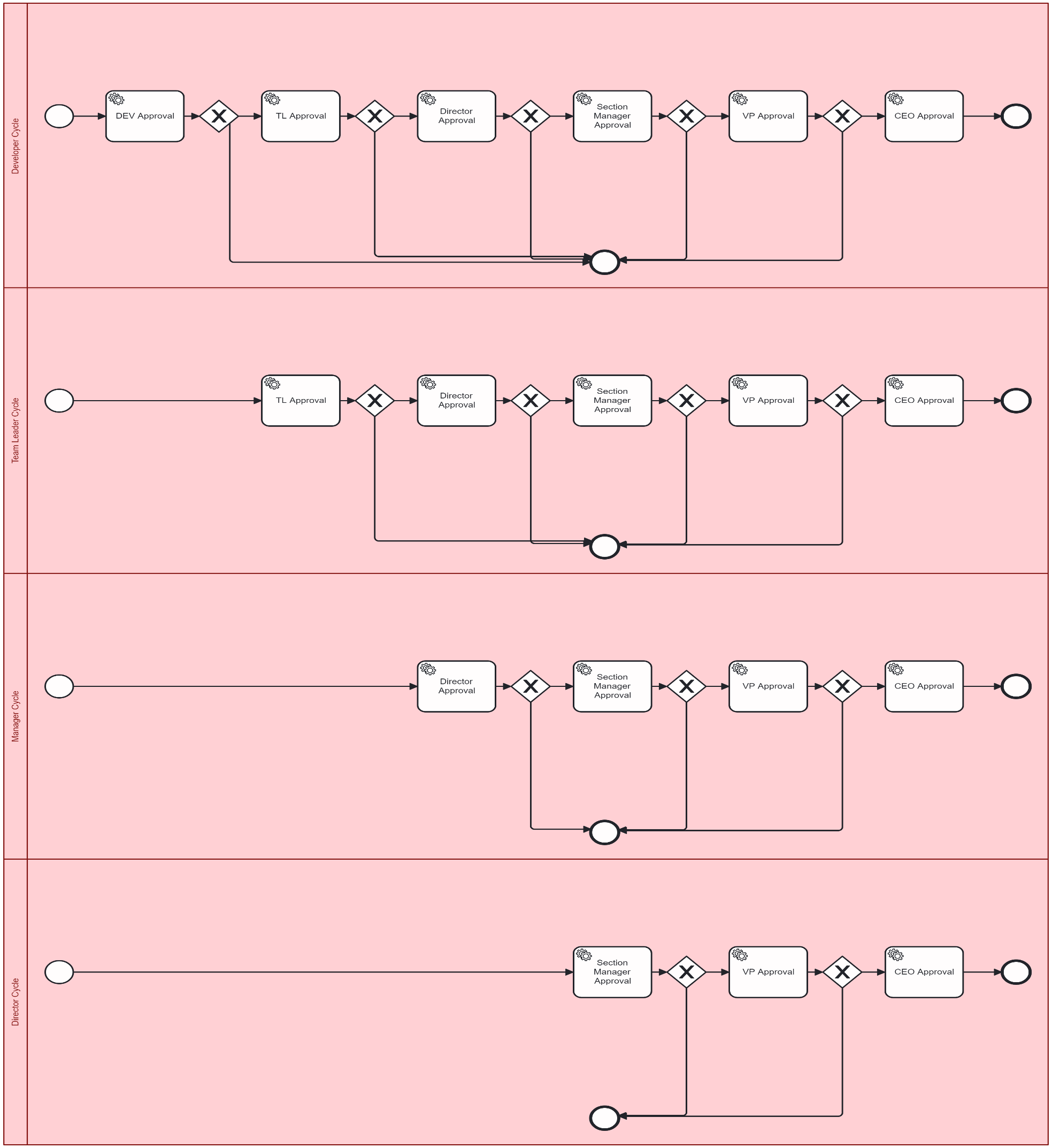
***Request WF Diagram***



This is a demo for the WF diagram for any request coming through all the above roles

DEV: Developer

TL: Team Leader

MGR: Manager

DIR: Director

VP: Vice president

CEO: Chief executive officer

every role has its own WF cycle to go through

task is a service task which call java delegate class to implements its business and log its data and send the request to the next step based on the action taken

steps:

1. Deployment:

once the application started the deployment of the WF will be done:

*/\*\*  
 \* this will deploy this diagram  
 \*/*@PostConstruct  
public void deployProcess() {  
 repositoryService.createDeployment()  
 .addClasspathResource("stc-emp-requests.bpmn")  
 .deploy();  
}

1. Create new request (start new process instance)

<http://localhost:8080/process/doAction/dev-Process>

<http://localhost:8080/process/doAction/tl-Process>

<http://localhost:8080/process/doAction/mgr-Process>

<http://localhost:8080/process/doAction/dir-Process>

this will return process instance id to be passed in the next step as this f5590a08-5c8a-11ef-bbee-00155d0ab4ad

1. Every role will fetch his inbox through this API

<http://localhost:8080/process/tasks?assignee=TL>

this will get list of String of process instance IDs that are active in every step based on the assignee

*/\*\*  
 \* get tasks for each role  
 \* @param assignee  
 \* @return  
 \*/*public List<String> getInboxByAssignee(String assignee) {  
 List<Task> tasks = taskService.createTaskQuery().taskAssignee(assignee).active().list();  
 return tasks.stream()  
 .map(task -> "Task ID: " + task.getId() + ", Name: " + task.getName())  
 .collect(Collectors.*toList*());  
}

1. Take approval or rejection action on the request

<http://localhost:8080/process/doAction/dev-Process>

<http://localhost:8080/process/doAction/tl-Process>

<http://localhost:8080/process/doAction/mgr-Process>

<http://localhost:8080/process/doAction/dir-Process>

body  
{

    "processInstanceId": "f5590a08-5c8a-11ef-bbee-00155d0ab4ad",

    "approved": **true**

}

*/\*\*  
 \* take action and move ticket to next step  
 \* @param processInstanceId  
 \* @param approved  
 \*/*public void takeAction(String processInstanceId, boolean approved) {  
 // Check if the process instance exists and is active  
 long count = runtimeService.createProcessInstanceQuery()  
 .processInstanceId(processInstanceId)  
 .active()  
 .count();  
  
 if (count == 0) {  
 throw new RuntimeException("Process instance with ID " + processInstanceId + " does not exist or has already ended.");  
 }  
  
 Map<String, Object> variables = new HashMap<>();  
 variables.put("approved", approved);  
 runtimeService.setVariables(processInstanceId, variables);  
  
 // Trigger the waiting task or event  
 runtimeService.createMessageCorrelation("ActionTaken")  
 .processInstanceId(processInstanceId)  
 .setVariable("approved", approved)  
 .correlate();  
}