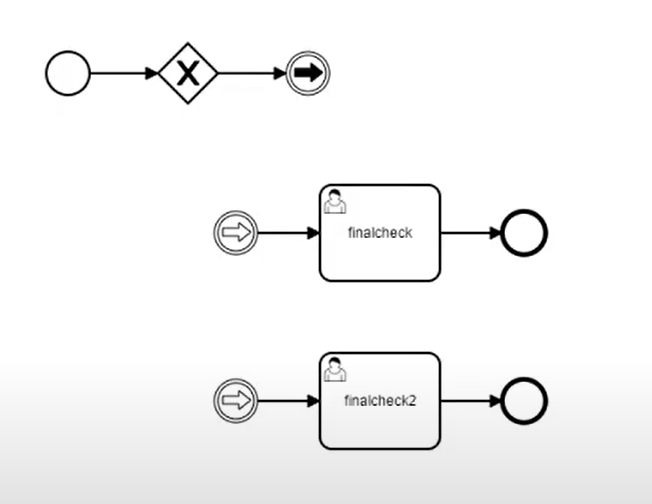
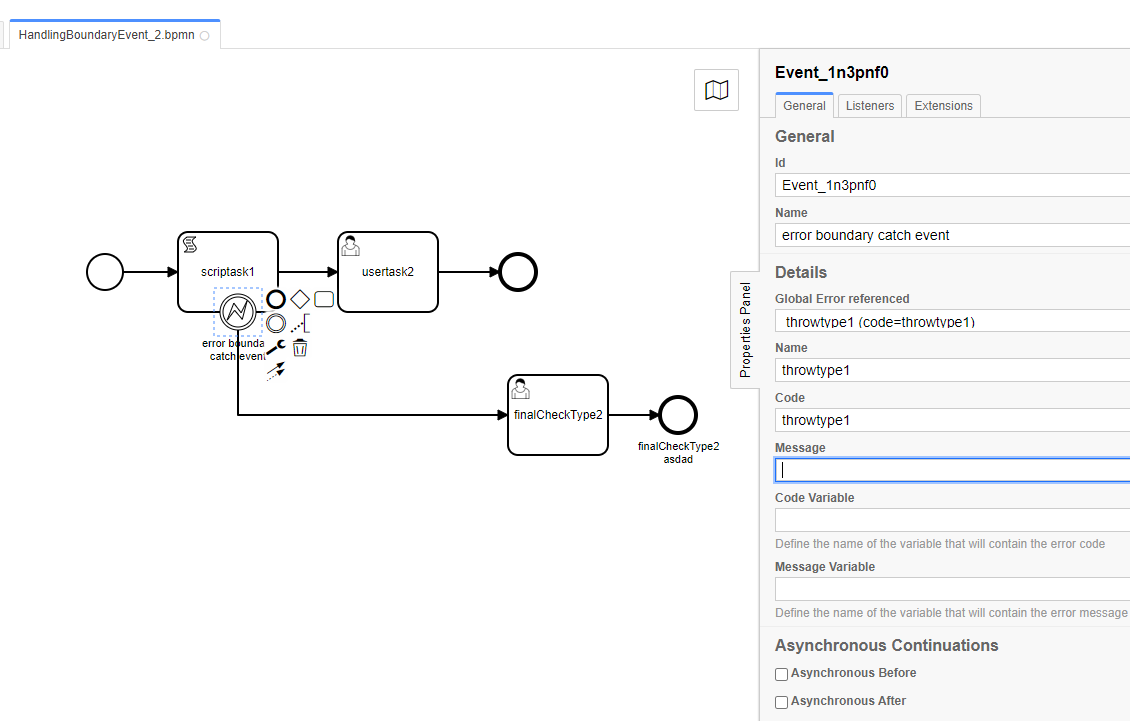
Error event:

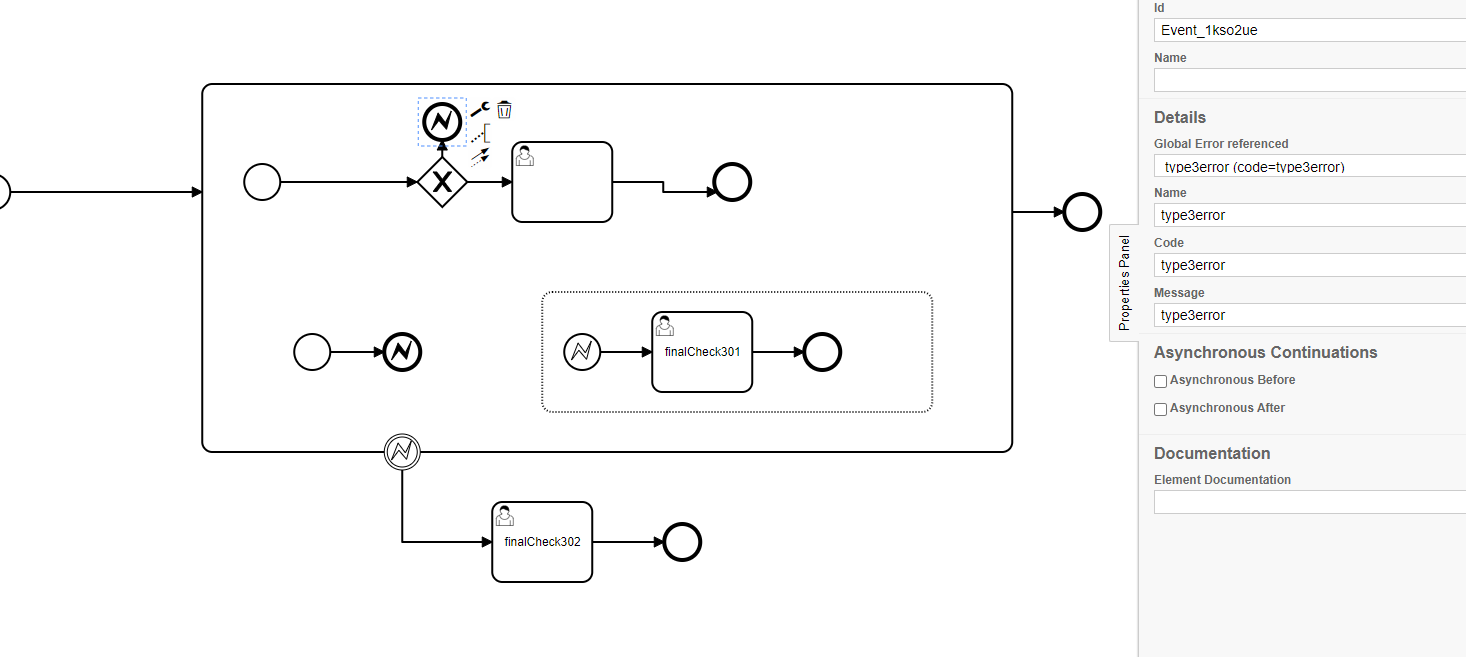
LinkIntermediateTHrowevent, LinkIntermediateCatchEvent



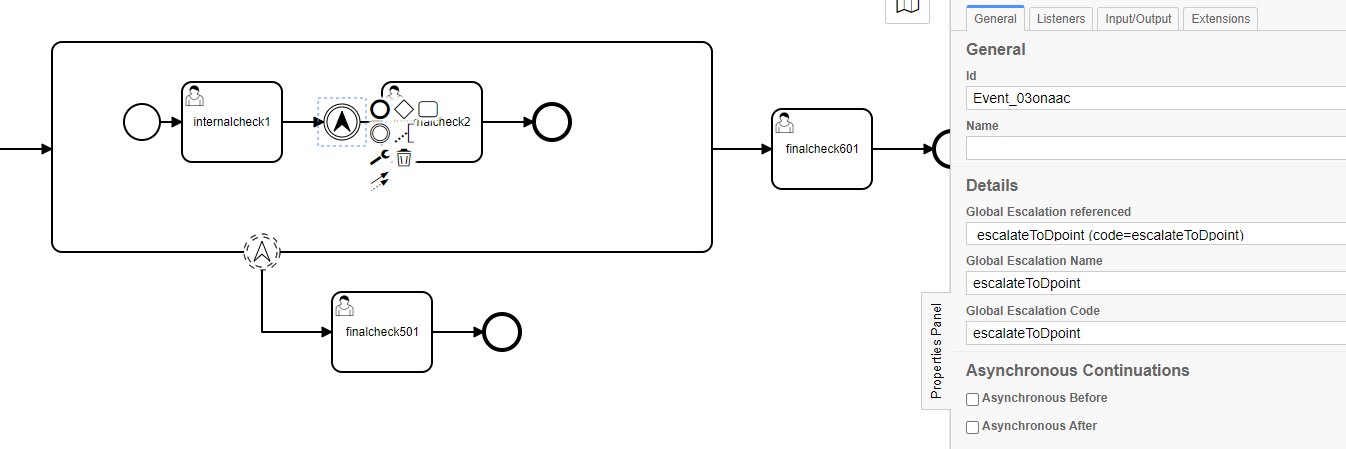
Error boundary catch event



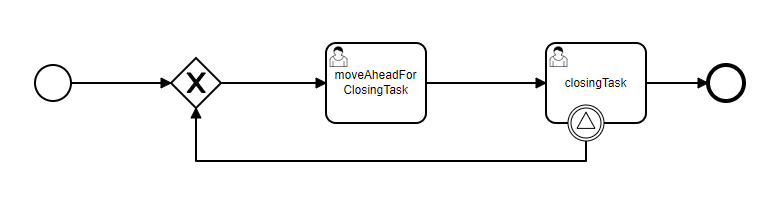
Event subprocess, and trycatch



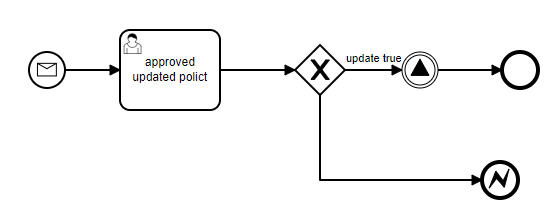
Escalation(throw and catch) event – executes 501 then flow from where the escalation was escalated continuous after the end of 501 flow



Signal catch event below



Signal throw event



**Business Process Life Cycle:**

1. **Design:** breakprocess into multiple tasks
2. **Model:** Model it using suitable BPM software
3. **Execute:** Execute process or put a system in place
4. **Monitor:** Monito and analyze the system
5. **Optimize:** Make changes to the process to improve it

Business Process Modelling: represents company’s business processes or workflows

Modelling techniques: BPMN, UML diagrams, flowcharts technique, data flow diagrams, role activity diagrams, Gnatt charts

BPM Tools:

**Commercial BPM**: Oracle BPM, IBM BPM, Pega BPM, ActivitiVOS

**Open Source BPM**: Camunda BPM, jBPMN, Flowable, Catify BPMN Engine, Activiti

**BPMN-CMMN-DMN:**

**BPMN –** precise, complete, graphical notation for documenting well defined process

Represents semantics of complex processes easily and in an intelligible form

Resolve many ambiguities found in textual process specifications

CMMN- graphical notation for capturing work methods

Less complex notation, better fit when describing knowledge intensive business cases

DMN – stand. Approach for descrbing and modelling repeatable decision within organisations

**Model and instance:** process instance is specific occurrence or execution of business process

Process instance can be independently monitored and managed

**CAMUNDA:** open source, decision automation platform,

Camunda bpm ships with tools for creating workflow and decision models

User can operate deployed models in prod and execute workflow tasks assigned to them.

Provides bpmn standard compliant workflow engine and dmn standard compliant decision engine, which can be embedded in java app and with other languages via rest.

Open source workflow engine

Light weight micro service workflow engine and decision automation platform

Supports cmmn and dmn notations in addition to BPMN 2.0

High performing workflow engine with effective persistence mechanism

Cloud native and multi tenant support

CDI/JAVA EE integration

**CAMUNDA getting started:**

Standalone – War deployment – Gaia K8 platform

Docker + GKP

Springboot Startup – Gaia Application Platform

Camunda cloud

**Camunda Setup:**

Create springboot maven project

Add camunda dependency

Update application.yml

Add process definition

Start application server

**Camunda features:**

Cockpit: tool for technical process operations

Task list: allows end users to work on assigned tasks, similar to user portals in pega

Optimize: analytics and reporting tool, similar to pega AES tool

Cawemo: collaborative modelling tool similar to designer studio for flow diagram creation

Admin: for application access control, allows users to manage camunda web application

**Camunda architecture:**

Enterprise

Open source

Operator

Task worker

Business Analyst/ Developer

Process owner

Administrator

Optimize

Admin

Cockpit

Custom app

Modeler

Tasklist

REST/JAVA

REST

Java/Rest API, Engine

File

Elastic Search

SQL DB

**Embedded Process Engine:**

**Container Managed Process Engine**

**Stand alone process Engine**

**Cluster Model**

**Shapes:**

Start shape,

end shape,

task – work that needs to be done, manual/automated, flavours like human,service,send

gateway – allow modelling decision based on data and events as well as fork/join concurrency

Intermediate/boundary:

Intermediary shapes can be used to pause flow execution and wait for trigger

Boundary shapes can be used for error handling/escalation/retry process steps etc

Human task – work needs to be done by human, process execution paused until action is taken

Script task – automated-

Script format support JSR-223 compatble lang like JS/Gtoovy/JRuby etc

Add corresponding jars in classpath

Heads-up-js uses nashorn engine which be deperecated in future java versions

Groovy is included in pre packaged camunda distributors

Call activity-automated

Invokes another BPMN process /CMMN/sub process

Allows transfer of data from parent to child process and vice versa

Business rule task – automated – calling a decision table in BPMN process

Service task – automated – invoke service and process the data

Send task – automated – used to send message-similar behaviour like service task -give visual rep

Receive activity – automated- wait state – used to make process wait until message arrives

Manual task – automated

Represent step outside of the process

Example person replacing the procduct at customer place

Shape used only for documenting purpose

**Gateways:**

* Parallel gateway: creates parallel path without checking condition,for incoming flows, it wait for all incoming flows before triggering flow through sequence flows
* Inclusive gateway: create alternate and parallel paths withing flow, all condition expression are evaluated unlike exclusive gateway
* Exclusive gateway: used to create alt path within process flow, diversion point, only one of the paths can be taken
* Event gateway: represent branching point in process where alt paths that follow gateway based on events

**Boundary Events:** attached to tasks, subprocess, or a call activities. They are waiting for event to happen while that activity is active. – Interrupting and non interrupting timer are example.

**Process Modelling life cycle:**

**Design Process model:** model process using camunda modeler, save the bpmn file in resource folder

**Code in IDE Modeler:** code your logic in intellij,by extending interface java delegate, put code directly in modeler through expressions or script

**Integrate:** integrate code using script, java class, delegate expression, listener tab

**Execute:** save process and run it from your choice of client

**Process Variables:** variables associated to ‘process instance’ and available throughout life cycle of process.

Camunda process engine offers JAVA API’s to set /get process variables which can be used in your custom java code or scripts via ‘Delegation code’

Global variables: available throughtout the process

Local variables: only during the execution of shape

Transient var: doesn’t get stored in DB and used only during execution

**Modeler – Properties Panel:**

**General tab**:

**Forms tab**:

**Variable’s tab**: available only for call activity, used to map process variables from parent to child process and vice versa

**Listener tab**: task listener is available only for human task

**Input/Output tab**: available on all shapes, inputs are typically passing data for reusable logic like complex calculation, process variable created for passed inputs will be ‘local’

Outputs are retrieved from business rule task or a reusable utility and mapped as ‘global variables’

**Field Injection tab**: available service task and send task, legacy implementation of Inputs/outputs, passing variables will not be persisted in DB

**Extension tab:** available on all shapes, used to store name/value pairs, only way to get these details is to read the BPMN xml file

**Execution Life Cycle:**

**Execution Listener:** Start and stop events

Task Listener: only in human task, create|update|timeout|delete

**Delegation code:** allows to execute external java code, scripts, evaluate expressions, when certain events occur during process execution

|  |  |  |
| --- | --- | --- |
| Delegation code type | Interface - Org.camunda.bpm.engine.delegate | BPMN Shape |
| Java Delegate | JavaDelegate | Service Task/Send Task |
| Delegate variable mapping | DelegateVariableMapping | Call activity |
| Execution listener | ExecutionListener | All Shapes |
| Task listener | TaskLIstener | Human task |

**Expression Language:** supports unified expression language(EL), specified as part of the JSP 2.1 standard ISR-245, uses open source JUEL implementation, some use cases given below and full list of supported functionality on the right

Seting process variable - ${execution.setVariable(‘variable’,’value’)}

Condition Evaluation on fork - ${!shipOrder!=true}

Setting assignee for human task - ${assignee}

Callling a method bean - ${mybean.calculateX()}

**Element templates:** templates are way to extend camunda modeler with domain specific diagram elements such as for example user tasks. Json file included in resource/element-templates folder in camunda modeler distribution

**Embed modeler into custom apps:**

Embed modeler into custom apps using Bpmn.js dmn.js, easily customize modeler to have custom validations/styles etc on bpmn.io

Generate BPMN/DMN/CMMN diagrams dynamically using fluence builder API:

Delegate vs expression vs Delegate expression

Camunda Events

Timer events:

1. Timer: events which are triggered by defined timer, time date, time duration, time cycle
2. Timer catch: interrupting event, when timer fires(e.g. after specified interval), sequence flow going out of timer intermediate event is followed
3. Timer boundary: non interrupting, these event leads to original activity not being interrupted, activity stays there

Message Events:

1. Message: action that refers to specific addressee and represents or contains info for the addressee is a message, can be used to start process
2. Message intermediate throw: sends message to external service, event has same behaviour as service task
3. Message catch: when token arrives at message intermediate catching event, it will wait there until message with proper name arrives
4. Message boundary event(interrupting): boundary events are catching events that are attached to activity, activity is interrupted and sequence flow going out of event is followed
5. Message boundary event(non-interrupting): catching events that are attached to an activity,
6. Message end event: process execution arrives at message end event, current path is ended and message is sent

Error events:

1. Error: events which are triggered by defined error, BPMN error, execution continuous from catching event

Escalation events:

1. Escalation: mostly used to communicate from sub process to upper process, unlike error , escalation event is non critical and execution continues at location of throwing

Signal events: event of global scope(broadcast) delivered to all active handlers

Cancel and compensation events: cancel end event can only be used in combination with transaction subprocess, when cancel end event is reached, cancel event is thrown which must be causght by cancel boundary event, cancel boundary event then cancels transaction and triggers compensation.

Conditional events: defines event which is triggered if given condition is evaluated to true.

**Camunda Engine Interface:**

Process Engine

Configuration- Camunda.cfg.xml

Process Engine

Task Service

Runtime Service

Repository Service

…

Identity Service

Management Service

History Service

Repository: offers operations for managing and manipulating deployments and process definitions

Runtime Service: deals with starting new process instances of process definitions

Task service: deals with quering, creating, manipulating, claiming tasks

Identity service: allows the management(creation, update, deletion, quering, etc) of groups and users

History service: exposes all historical data gathered by engine

Management service: allows to retrieve info about database tables and table metadata

Decision service: allows to evaluate decisions that are deployed to engine

Database: RE\_ : static info like definiotions and resources

RU\_: runtime , ID\_: identity -users,groups , HI\_: history tables, GE\_: general data

DMN: decision modelling standard defined and owned by OMG group, industry standards for modelling and executing decisions that are determined by business rules, camunda used MDN standards for modelling decision, easy visualize and read for business and it groups,

Hit policy:

|  |  |  |  |
| --- | --- | --- | --- |
| Hit policy | Description | Synopsis | Accepted #of satisfied results |
| U | unique | No overlapping rules | 1 |
| F | first | Rule order | 1 |
| P | Priority | Results by priority |  |
| A | Any | Overlapping inputs can match same output results by rule order | 1 |
| C | Collect | Returns all hits in arbitrary order, aggression of output rules is allowed | >1 |
| R | Rule order | Returns all this in rule order | >1 |
| O | Output order | Returns all this in decreasing output priority order |  |

REST API’s:

Open api complaint -starting 7.13

Generate clients in your preferred language

Rest api is an artifact of its own, which means that it can be embedded in any other JAX-RS app independent of the engine, it has been tested with jersey, resteasy, wink

Rest apoi are just wrappers around the camunda java interfaces like taskservice/ identity service/history service etc

Provides interfaces to extend authentication

Provides comprehensive set of API’s for BPMN / DMN/CMMN and follows HATEOAS architecture via HAL

Dependency: camunda-bpm-spring-boot-starter-rest

Camunda springboot starter rest has jersey integrated which makes it easy to extend the camunda rest API’s and add custom api

**Applications, Forms, and SDK:**

Camunda OOTB task list/cockpit/admin portals are built in angular(specially angular 1.7)

All these apps needs stick sessions/statefulness, causes lot of overhead in terms of maintenance as we need different implementation

Camunda provides form sdk which can be integrated with any JS based app and is written in JS

Features include form handling/variable handling/script handling/angular js integration

Form SDK provides client to connect to engine needs jQuery dependency

Generate form:

Embedded form:

External form:

Enterprise features: Process Heat maps, decision table execution history, process instance modification, operational metrics, completed process instance history, optimize tool, cawemo, support

Data Extraction:

Camunda provides various levels of history: none, activity, audit, full, auto

Options: nightly job, spring eventing bridge

Camunda.bpm.eventing.execution=true

Camunda.bpm.eventing.hiostory=true

Camunda.bpm.eventing.task=true