Short introduction to BPMN 2.0

Business Process Model and Notation has been developed to provide a standard for business process models in order to make process depictions understandable not only for process modelers but also for implementers. The process diagrams are based on symbols for tasks, sub-processes, call activities events and more. The notation is based on symbols which are connected by directed arrows. These symbols can be distinguished based on their shape and content - for example, timer events have a clock on them and abort events have a cross on them[1]. Start and end events are simple - start events are blank thin lined circles, end events are displayed by a thick black circle. Start events are used to indicate the start of a process model, end events for the end of a process. In between as already mentioned, elements can be arranged and connected with arrows in an order that corresponds to the real workflow. An example for a simple workflow is a Gargle Covid-19 Test procedure similar to the process provided for people living and working in Vienna.

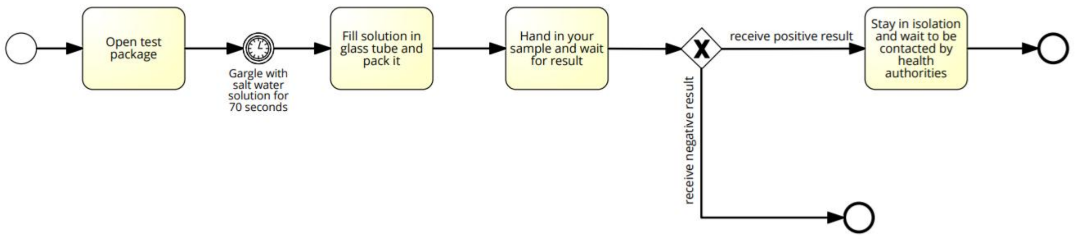
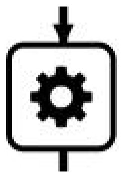

Open test package
Gargle with salt water solution for 70 seconds
Fill solution in glass tube and pack it
Hand in your sample and wait for result
receive positive result
Stay in isolation and wait to be contacted by health authorities
receive negative result

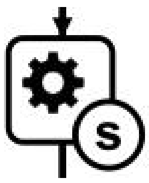
Figure 1: Example for Gargle Covid-19 Test Procedure

The person taking the test needs to open the package with the test utensils. Then the manual tells you to gargle with a salt solution for about 70 seconds. After that you fill the solution in a glass tube, wrap it up and deliver the sample to a super market. Then you need to wait for the result - in case it is negative you can proceed with a normal routine. Otherwise with a positive test result it is advised to stay in self-isolation and wait for the local health authorities to contact you for further instructions.

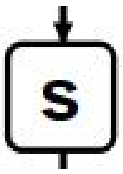
Workflow engines are applications which allow the execution of workflows. Many support BPMN and allow the execution of processes modeled with this standard. The one we want to use is the [redacted] also called [redacted]. [redacted] is also based on BPMN but uses extensions for a wider functionality. Thus it has been used in research projects with focus on manufacturing processes. We use the Workflow engine mainly for orchestration of manufacturing processes. [redacted] can send requests via OPCUA, HTTP, etc. The endpoints for these requests can be stored as additional variables for the process. Thus interoperability is a great advantage for a process with various components. [redacted] has been implemented with specific extensions to BPMN. In addition to the endpoints, other variables, which may contain initial data but can also be overwritten during the process, can be defined by the user. The user can integrate tasks such as



Service Call (HTTP Request)



Service Call with Script (HTTP Request)



Script

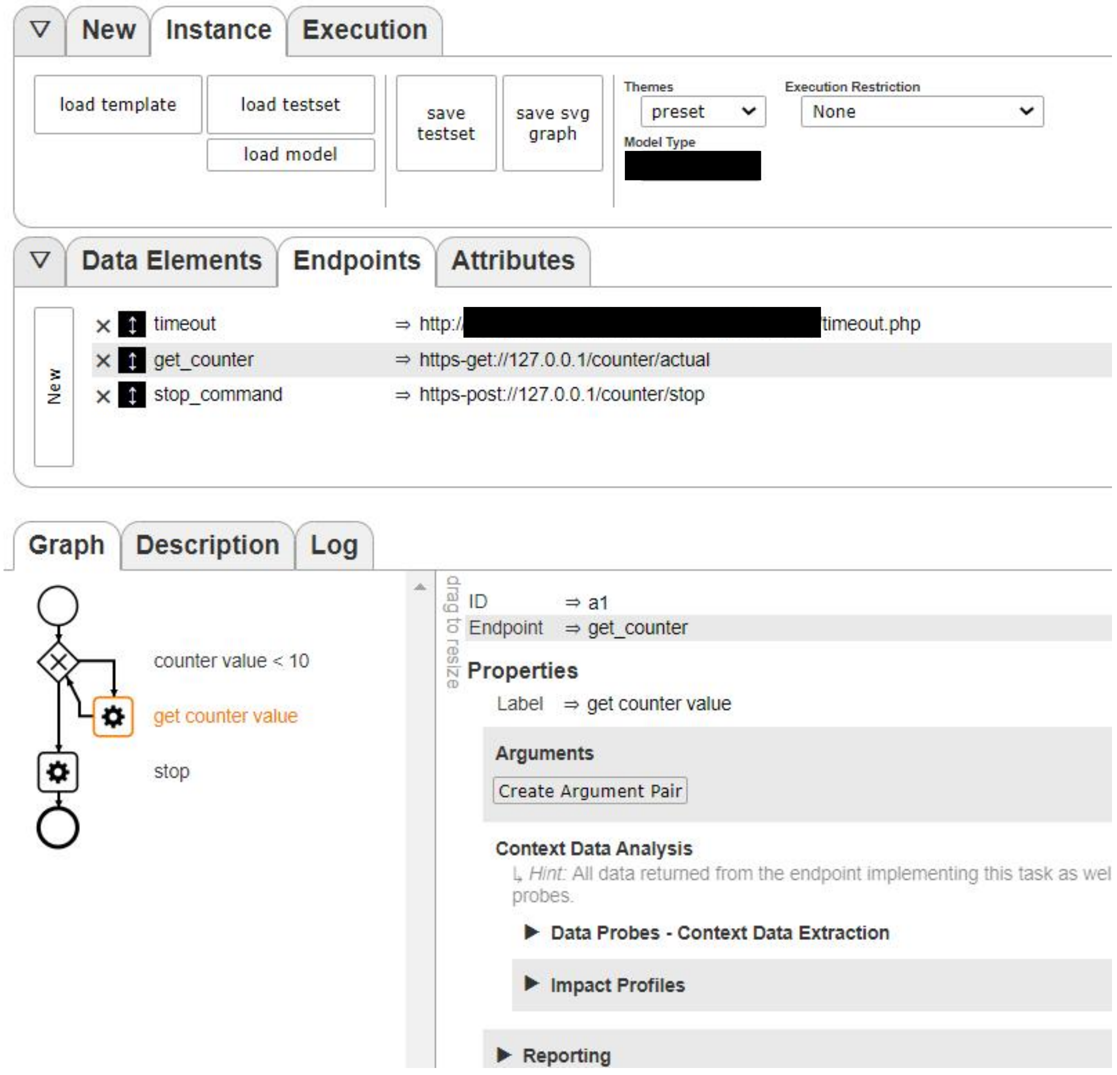
[1] For more detailed information, see http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf

Service call
Service call allows the user to integrate a service call with a defined endpoint. For example, the user may define an URL for accessing the variable provided by an HTTP server using a GET request. To receive the data, the user can integrate a service call into their workflow and define the call method (GET) along with the endpoint address.
get_temperature => https-get://<server_ip>/process1/values/temperature

Service Call with Script
This symbol works similar to Service Call but adds a script to process the received data.

Script
Script adds a simple script to the workflow which can be used to process a variable and use the output in the following tasks of the process.

A simple process model in [redacted][2] is shown in the following figure:



Themes: preset
Execution Restriction: None
Model Type: [redacted]

timeout	⇒ http:/[redacted]timeout.php
get_counter	⇒ https-get://127.0.0.1/counter/actual
stop_command	⇒ https-post://127.0.0.1/counter/stop

ID ⇒ a1
Endpoint ⇒ get_counter

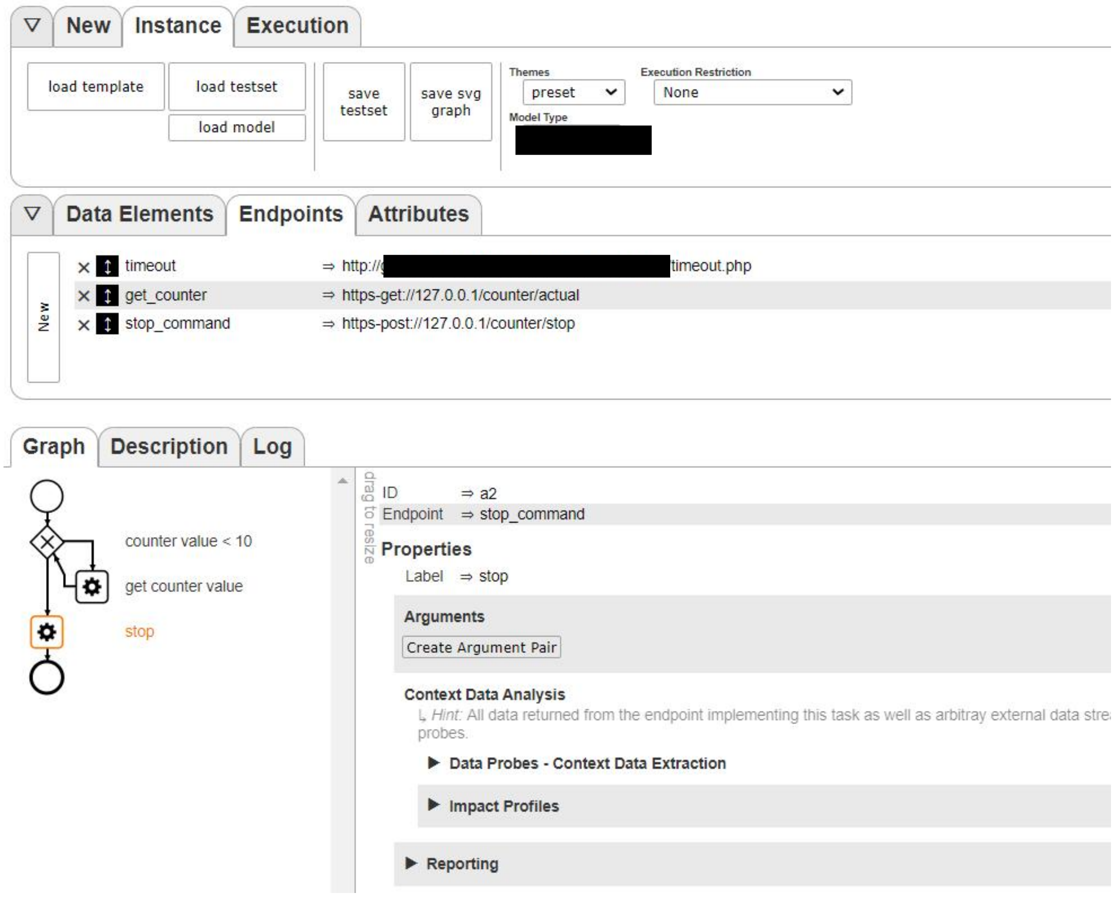
Properties
Label ⇒ get counter value

Arguments

Context Data Analysis
↳ *Hint:* All data returned from the endpoint implementing this task as wel probes.

Figure 2: Attributes for Service Call 'get counter value'

[2] See https://[redacted]



timeout	⇒ http://[redacted]timeout.php
get_counter	⇒ https-get://127.0.0.1/counter/actual
stop_command	⇒ https-post://127.0.0.1/counter/stop

ID ⇒ a2
Endpoint ⇒ stop_command

Properties

Label ⇒ stop

Arguments

Create Argument Pair

Context Data Analysis

↳ *Hint:* All data returned from the endpoint implementing this task as well as arbitray external data stre
probes.

▶ Data Probes - Context Data Extraction

▶ Impact Profiles

▶ Reporting

Figure 3: Attributes for Service Call 'stop'

As soon as the counter reaches 10 the loop is terminated and a stop command is sent via a POST request.