GEC21 - OEDL Tutorial 1

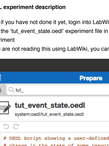
Overview

This first part shows how to design, execute, and view the results of an experiment, which dynamically reacts to a user-defined event.

This experiment demonstrate OEDL's capability to allow user to define events which will trigger based on the state of the used resources. In other words, if one or more resources reaches a specific state, the event will trigger and some user-defined tasks will be executed.

In this experiment:

- we start with 4 resources, i.e. 2 initial 'workers' and 2 backup ones
- all workers have a ping application associated to them
- the 2 initial workers starts their ping applications
- we define a custom event which will trigger when a running ping application is stopped. The Experiment Controller will periodically monitor the state of all resources to check for this condition
- furthermore, we define a set of tasks to execute if the event is triggered. In this case, the task is to start a new ping application on one of the backup resources.
- every 20 seconds, we purposely stop the ping application running on one of the initial workers
- we display a graph of the ping's RTT for each of the 4 resources and observe that a new ping instance starts when a previously running one is stopped



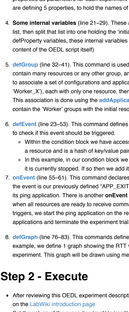
Experiment 1 Overview

Step 1 - Design/Setup

For specific help on using LabWiki, please refer to the LabWiki introduction page

The OEDL experiment description

- First, if you have not done it yet, login into LabWiki
- Load the 'tut_event_state.oedl' experiment file in the 'Prepare' Panel of LabWiki. This file contains the OEDL script for this 1st experiment
- If you are not reading this using LabWiki, you can view this OEDL file online at: http://git.io/xr4pag



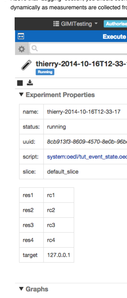
Experiment 1 OEDL Extract

Walk-through the OEDL experiment description

1. First, a reminder that all details on OEDL are available in the OEDL reference page
2. loadOEDL (line 12). This command is used to include in your OEDL experiment other external OEDL scripts. In this example, we are loading the definition of a *ping* application, which has been instrumented with OML
3. defProperty (line 14–18). This command is used to define experiment properties (aka variables), you can set the values of these properties as parameters for each experiment trials, and access them throughout the entire experiment run. In this example, we are defining 5 properties, to hold the names of each of the resources that we will use and the target for the ping application.
4. **Some internal variables** (line 21–29). These are classic simple Ruby commands that allow us *put all our resource in a single* list, then split that list into one holding the 'initial' resources, and one holding the 'backup' resources. As opposed to the above defProperty variables, these internal variables cannot be set at the start of each experiment trial (without having to change the content of the OEDL script itself).
5. defGroup (line 32–41). This command is used to define a group of resources which we will use in this experiment. A group may contain many resources or any other group, and a resource may be included in many groups. This commands may also be used to associate a set of configurations and applications to all resources in a group. In this example, we first define 4 groups (e.g. 'Worker_X'), each with only one resource, then we are associating an instrumented ping to the unique resource in each group. This association is done using the addApplication. Furthermore, we also define a final group ('Initial_Worker'), which will contain the 'Worker' groups with the initial resources.
6. defEvent (line 23–53). This command defines the name of a user's custom event and the block of conditions which will be used to check if this event should be triggered.
   - Within the condition block we have access to the 'state' variable, which holds a array. Each element of that array represents a resource and is a hash of key/value pairs corresponding to each properties of that resource.
   - In this example, in our condition block we check for each resource if it failed before. If not we check if is an application and if it is currently stopped. If so then we add it to the list of failed resource, and we trigger the event.
7. onEvent (line 55–61). This command declares the set of actions to perform when a specific event is triggered. In this example, in the event's our previously defined "APP_EXITED". The actions to perform in this case is to select a backup resource and start its ping application. There is another **onEvent** declaration further (line 68–74), for the event "APP_UP_AND_INSTALLED", i.e. when all resources are ready to receive commands and all applications associated to them are installed. When this event triggers, we start the *ping* application on the resources within the 'Initial_Worker' group, then after 60 seconds we stop all applications and terminate the experiment trial.
8. defGraph (line 76–83). This commands defines the graphs that will be displayed while the experiment trial is running. In this example, we define 1 graph showing the RTT values from the ping applications against time for each resources in our experiment. This graph will be drawn using measurements enabled in the previous defGroup blocks.

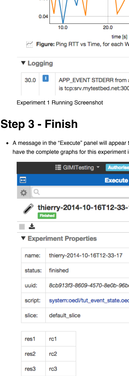
Step 2 - Execute

- After reviewing this OEDL experiment description, drag-and-drop it from the "Prepare" panel to the "Execute" panel, as described on the LabWiki introduction page
- Set the values of the properties 'res1' to 'res4' to the names of your allocated resources. Similarly set the 'Slice' property to your own slice. (You can optionally decide to give a name to your experiment, if not LabWiki will assign a default unique name to it.)
- Click on the "Start Experiment" button. You will soon see output messages under the "Logging" section. Some of these messages are from the OMF Experiment Controller, which interprets your OEDL experiment description and sends corresponding commands to the resources. Other messages are from the resources themselves (either the VM nodes or the applications), reporting on configuration and command results.



Experiment 1 Execute Screenshot

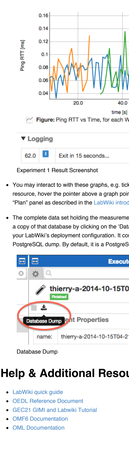
- Above that "Logging" section, you should soon see the graph, which we defined in the OEDL experiment description. It is drawn dynamically as measurements are collected from the resources.



Experiment 1 Running Screenshot

Step 3 - Finish

- A message in the "Execute" panel will appear to inform you that the experiment execution has finished. At this stage, you should have the complete graphs for this experiment in that panel, which should look as follows.



Experiment 1 Result Screenshot

- You may interact to with these graphs, e.g. tick or un-tick the legend's keys to display only results from the first or/and second resource, hover the pointer above a graph point to display the underlying data point, drag-and-drop the graph via its icon to the "Plain" panel as described in the LabWiki introduction page
- The complete data set holding the measurements collected from this experiment is stored in an SQL database. You can retrieve a copy of that database by clicking on the 'Database Dump' button in the 'Execute' panel. The format of that copy is depends on your LabWiki's deployment configuration. It could be an iRODS dump, a Zipped archive of CSV files, a SQLite3 dump or a PostgreSQL dump. By default, it is a PostgreSQL dump.

Database Dump

Help & Additional Resources

- LabWiki quick guide
- OEDL Reference Document
- GEC21 GIMI and Labwiki Tutorial
- OMF6 Documentation
- OML Documentation