GEC21 - OEDL Tutorial 3

Overview

This third part shows how to design, execute, and view the results of an experiment, which dynamically retrieves the resources available in the experimenter's slice and select a subset to use in its execution.

This experiment demonstrate OEDL's integration with the Slice Service of the Labwiki workspace, which allows user to dynamically retrieve the list of resources available in their slice and seamlessly select and use some of these resources in their experiment.

In this experiment:

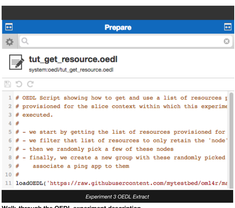
- we start by getting the list of resources provisioned for our slice
- we filter that list of resources to only retain the 'node' ones
- then we randomly pick a few of these nodes
- finally, we create a new group with these randomly picked nodes and associate a ping app to them

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_get_resource.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/43sLkA



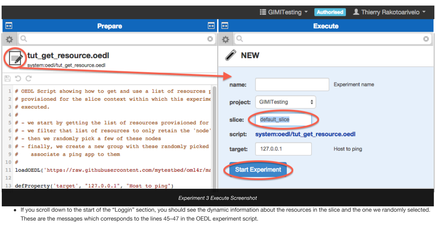
Experiment 3 OEDL Extract

Walk-through the OEDL experiment description

1. First, a reminder that all details on OEDL are available in the OEDL reference page
2. **loadOEDSL** (line 11). This command includes in your OEDL experiment other external OEDL scripts. In this example, we are loading the definition of a ping application (instrumented with OML)
3. **defProperty** (line 13). This command defines 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 1 property, which holds the target for the ping application.
4. **getResources** (line 41) This command retrieves the list of resources which are associated with the current slice used by the experimenter. It returns an array where each element is a resource record. Such a record is a hash describing the resource. The experiment script can then filter the list of resources based on specific keys and values in such a record (e.g. only retain the resources of type 'node')
5. **Some internal variables** (line 42–47). These are simple Ruby commands that select a random subset of only 'node' type resources, and print some information about them. As opposed to the above defProperty variables, internal variables cannot be set at the start of each experiment trial (without having to change the content of the OEDL script itself)
6. **defGroup** (line 49-54). 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 define a single group 'Random_Workers' including the previously selected random resources, then we associate an instrumented ping application to them. This association is done using the **addApplication**.
7. **onEvent** (line 56-62). This command declares the set of actions to perform when a specific event is triggered. In this example, the event is "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 resource within the 'Random_Workers' group, then after 40 seconds we stop all applications and terminate the experiment trial.
8. **defGraph** (line 64–71). 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.

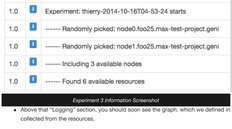
Step 2 - Execute

- 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 'Slice' property to your own slice's name. (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.



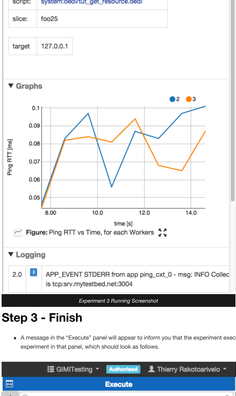
Experiment 3 Execute Screenshot

- If you scroll down to the start of the "Loggin" section, you should see the dynamic information about the resources in the slice and the one we randomly selected. These are the messages which corresponds to the lines 45-47 in the OEDL experiment script.



Experiment 3 Information Screenshot

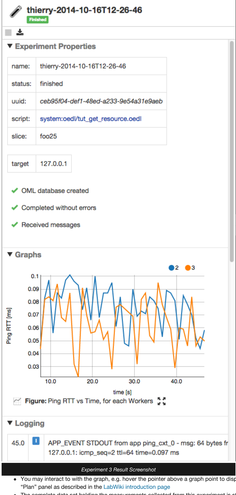
- Above that "Logging" section, you should soon see the graph, which we defined in the OEDL experiment description. It is drawn dynamically as measurements are collected from the resources.



Experiment 3 Running Screenshot

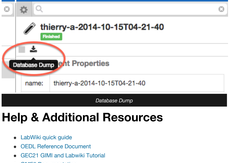
Step 3 - Finish

- 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 graph for this experiment in that panel, which should look as follows.



Experiment 3 Result Screenshot

- You may interact to with the graph, e.g. hover the pointer above a graph point to display the underlying data point, drag-and-drop the graph via its icon to the "Plan" 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' buttom in the 'Execute' panel. The format of that copy is depends on your LabWiki's deployment configuration. It could be an #ODS 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