Help Topic: Scripted Pokes to the MOOSDB

Spring 2020

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

Scripted Pokes to the MOOSDB

Here we cover how to have a script of pre-arranged pokes to the MOOSDB. This may be useful for a number of reasons besides debugging. The primary tool described here is the uTimerScript MOOS application. It is capable of (a) scripted pokes at pre-defined times after launch, (b) pokes having a poke-time specified to fall randomly within a specified interval, (c) pokes having numerical values falling with a uniformly random interval, and several other features including conditioning the running of the script based on other MOOS variables.

Where to get more information:

- uTimerScript: http://oceanai.mit.edu/ivpman/apps/uTimerScript

Or simply:

```
$ uTimerscript --web
```

Basic uTimerScript Usage

uTimerScript is configured with its own block in the MOOS configuration file. The general format is below. The primary entries are the events themselves, defined by a MOOS variable, value, and time or time-range when the event is to occur. There are many options for configuring the script. These options are described in the documentation, but a quick look at the options can be seen by typing uTimerScript --example on the command line.

```
ProcessConfig = uTimerScript
{
  event = var=<MOOSVar>, val=<value>, time=<value>
  event = var=<MOOSVar>, val=<value>, time=<value>
  ...
  event = var=<MOOSVar>, val=<value>, time=<value>

  [OPTIONS]
}
```

A Simple Example with uTimerScript

The below mission file contains a uTimerScript script for repeatedly posting the variable COUNTER_A with values 1-10 in ascending order roughly once every half second. The last event in the script is posted at time chosen from a random five second interval.

Listing 0.1: A simple counter example with uTimerScript.

```
// (wget http://oceanai.mit.edu/2.680/examples/utscript.moos)
ServerHost = localhost
ServerPort = 9000
Community  = alpha

ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB       @ NewConsole = false
  Run = uXMS         @ NewConsole = true
  Run = uTimerScript @ NewConsole = false
}

ProcessConfig = uXMS
{
  VAR            = COUNTER_A, DB_CLIENTS, DB_UPTIME
  COLOR_MAP      = COUNTER_A, red
  HISTORY_VAR    = COUNTER_A
}

ProcessConfig = uTimerScript
{
  paused = false

  event  = var=COUNTER_A, val=1,  time=0.5
  event  = var=COUNTER_A, val=2,  time=1.0
  event  = var=COUNTER_A, val=3,  time=1.5
  event  = var=COUNTER_A, val=4,  time=2.0
  event  = var=COUNTER_A, val=5,  time=2.5
  event  = var=COUNTER_A, val=6,  time=3.0
  event  = var=COUNTER_A, val=7,  time=3.5
  event  = var=COUNTER_A, val=8,  time=4.0
  event  = var=COUNTER_A, val=9,  time=4.5
  event  = var=COUNTER_A, val=10, time=5:10

  reset_max  = nolimit
  reset_time = all-posted
}
```

The mission may be launched from the command-line with:

```
 $ pAntler utscript.moos
```

This should open a new console window for uXMS displaying the variables COUNTER_A variable repeatedly

incrementing from 1 to 10. Note that reaching 10 happens somewhere between 0.5 and 5.5 seconds after reaching 9.

Exercises

Your goals in this part are:

1. Create a copy of the example mission file in Listing 1 above and save it locally.
2. Launch the mission. It should open a uXMS window. Follow the progress of the counter script.

```
$ pAntler utscript.moos
```

3. Take a look at the uTimerScript documentation linked from the web page. In particular, Section 3 Script Flow Control. Configure the script such that is paused when uTimerScript is launched. Launch the same mission and confirm that the script is initially not running. Then use uPokeDB to un-pause the script, and confirm it is running. Hint: to un-pause the script with uPokeDB, you'll need to know which variable to poke, and this also can be found in the uTimerScript documentation or by typing `uTimerScript -i` on the command line.
4. This is a bit of a pAntler exercise. Configure your mission to launch two versions of the script, the second version publishing to COUNTER_B. Note you will need two configuration blocks, each with a unique name. And you will need to launch uTimerScript twice within the Antler block, each with an alias. Hint: see the pXRelay example at the end of Lab 3.
5. Confirm your new mission launches and executes the two separate scripts and both counters are incrementing.
6. Configure the second script with a condition parameter. See Section 3.2 of the uTimerScript documentation. Use a condition such as `"condition = COUNTER_A > 5"`. Re-launch your mission. Confirm that the second script is paused periodically based on the state of the first script.
7. Add the pLogger application to your mission. You will need to add a pLogger entry to your ANTLER configuration block, and add the following pLogger configuration block at the end of your file.

```
ProcessConfig = pLogger
{
  AsyncLog = true
  WildCardLogging = true
  WildCardOmitPattern = *_STATUS
}
```

   Re-run the mission. Confirm that you see the pLogger application listed in the DB_CLIENTS variable in the uXMS scope.
8. Verify that a log file has been created. Since we didn't specify a name for the log file, by default it should be in a subdirectory of where you launched the mission, looking something like `MOOSLog_11_21_2024_____11_31_13/`. Enter the directory and confirm that you see an `.alog` file.
9. Take a look a the file by typing `more filename.alog`. Then take at look at the COUNTER variables using aloggrep (substituting of course the name of your `.alog` file:

```
$ aloggrep COUNTER_A COUNTER_B   MOOSLog_11_23_2016_____11_31_13.alog
```