uPokeDB: Poking the MOOSDB from the Command Line

June 2018

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

1 Overview

The uPokeDB application is a lightweight process that runs without any user interaction for writing to (poking) a running MOOSDB with one or more variable-value pairs. It is run from a console window with no GUI. For example, the alpha example mission is normally kicked off by hitting the DEPLOY button. The same could be accomplished from the terminal with:

```
$ uPokeDB alpha.moos DEPLOY=true, MOOS_MANUAL_OVERIDE=false
```

After accepting variable-value pairs from the command line, uPokeDB connects to the MOOSDB, displays the variable values prior to poking, performs the poke, displays the variable values after poking, and then disconnects from the MOOSDB and terminates. It also accepts a .moos file as a command line argument to grab the IP and port information to find the MOOSDB for connecting. Other than that, it does not read a uPokeDB configuration block from the .moos file.

Other Methods for Poking a MOOSDB

There are few other MOOS applications capable of poking a MOOSDB. The uMS (MOOS Scope) is an application for both monitoring and poking a MOOSDB. It is substantially more feature rich than uPokeDB, and depends on the FLTK library. The iRemote application can poke the MOOSDB by using the CustomKey parameter, but is limited to the free unmapped keyboard keys, and is good when used with some planning ahead. The latest versions of uMS and iRemote are maintained on the Oxford MOOS website. The uTermCommand application is a tool primarily for poking the MOOSDB with a pre-defined list of variable-value pairs configured in its .moos file configuration block. The user initiates each poke by entering a keyword at a terminal window. Unlike iRemote

it associates a variable-value pair with a key *word* rather than a keyboard key. The uTimerScript application is another tool for poking the MOOSDB with a pre-defined list of variable-value pairs configured in its `.moos` file configuration block. Unlike uTermCommand, uTimerScript will poke the MOOSDB without requiring further user action, but instead executes its pokes based on a timed script. The uMOOSPoke application, written by Matt Grund, is similar in intent to uPokeDB in that it accepts a command line variable-value pair. uPokeDB has a few additional features described below, namely multiple command-line pokes, accepting a `.moos` file on the command-line, and a MOOSDB summary prior and after the poke.

2 Command-line Arguments of uPokeDB

The command-line invocation of uPokeDB accepts two types of arguments - a `.moos` file, and one or more variable-value pairs. The former is optional, and if left unspecified, will infer that the machine and port number to find a running MOOSDB process is `localhost` and port 9000. The uPokeDB process does not otherwise look for a uPokeDB configuration block in this file. The variable-value pairs are delimited by the '=' character as in the following example:

```
$ uPokeDB alpha.moos FOO=bar TEMP=98.6 MOTTO="such is life" TEMP_STRING:=98.6
```

Since white-space characters on a command line delineate arguments, the use of double-quotes must be used if one wants to refer to a string value with white-space as in the third variable-value pair above. The value *type* in the variable-value pair is assumed to be a double if the value is numerical, and assumed to be a string type otherwise. If one really wants to poke with a string type that happens to be numerical, i.e., the string "98.6", then the ":=" separator must be used as in the last argument in the example above. If uPokeDB is invoked with a variable type different than that already associated with a variable in the MOOSDB, the attempted poke simply has no effect.

3 MOOS Poke Macro Expansion

The uPokeDB utility supports macro expansion for timestamps. This may be used to generate a proxy posting from another application that uses timestamps as part of it posting. The macro for timestamps is `@MOOSTIME`. This will expand to the value returned by the MOOS function call `MOOSTime()`. This function call is implemented to return UTC time. This following is an example:

```
$ uPokeDB file.moos FOOBAR=color=red,temp=blue,timestamp=@MOOSTIME
```

The above poke would result in a posting similar to:

```
FOOBAR = color=red,temp=blue,timestamp=10376674605.24
```

As with other pokes, if the macro is part of a posting of type double, the timestamp is treated as a double. The posting

```
$ uPokeDB file.moos TIME_OF_START=@MOOSTIME
```

would result in the posting of type double for the variable TIME_OF_START, assuming it has not been posted previously as a different type.

4 Providing the ServerHost and ServerPort on the Command Line

The specification of a MOOS file on the command line is optional. The only two pieces of information uPokeDB needs from this file are (a) the server_host IP address, and (b) the server_port number of the running MOOSDB to poke. These values can instead be provided on the command line:

```
$ uPokeDB FOO=bar --host=18.38.2.158  --port=9000
```

If the host or the port are not provided on the command line, and a MOOS file is also not provided, the user will be prompted for the two values. Since the most common scenario by convention has the MOOSDB running on the local machine ("localhost") with port 9000, these are the default values and the user can simply hit the return key.

```
$ uPokeDB FOO=bar                       // User launches with no server host/port info
$ Enter Server: [localhost]             // User accepts default by hitting Return key
$    The server is set to "localhost"   // Server host confirmed to be set to "localhost"
$ Enter Port: [9000] 9123               // User overrides the default 9000 port with 9123
$    The port is set to "9123"          // Server port confirmed to be set to "9123"
```

5 Session Output from uPokeDB

The output in Listing 1 shows an example session when a running MOOSDB is poked with the following invocation:

```
$ uPokeDB alpha.moos DEPLOY=true RETURN=true
```

Lines 1-16 are standard output of a MOOS application that has successfully connected to a running MOOSDB. Lines 19-23 indicate the value of the variables prior to being poked, along with their source, i.e., the MOOS process responsible for publishing the current value to the MOOSDB, and the time at which it was last written. The time is given in seconds elapsed since the MOOSDB was started. Lines 26-30 show the new state of the poked variables in the MOOSDB after uPokeDB has done its thing.

Listing 5.1: An example uPokeDB session output.

```
1  ------------------------------------------------
2  |       This is an Asynchronous MOOS Client    |
3  |       c. P. Newman U. Oxford 2001-2012       |
4  ------------------------------------------------
5
6  ---------------MOOS CONNECT----------------------
7    contacting a MOOS server localhost:9000 -  try 00001
8    Contact Made
9    Handshaking as "uPokeDB"........ [OK]
10 ------------------------------------------------
```

```
11
12  uPokeDB is Running:
13   +Baseline AppTick   @ 5.0 Hz
14   +Comms is Full Duplex and Asynchronous
15   +Iterate Mode 0 :
16     -Regular iterate and message delivery at 5 Hz
17
18
19  PRIOR to Poking the MOOSDB
20    VarName               (S)ource    (T)ime      VarValue
21    ----------------      ----------  ----------   -------------
22    DEPLOY                 uTimerScript1.92        "true"
23    RETURN                 pHelmIvP    1           "false"
24
25
26  AFTER Poking the MOOSDB
27    VarName               (S)ource    (T)ime      VarValue
28    ----------------      ----------  ----------   -------------
29    DEPLOY                 uPokeDB     22.58       "true"
30    RETURN                 uPokeDB     22.58       "false"
```

6 Publications and Subscriptions for uPokeDB

Variables published by the uPokeDB application

- `USER-DEFINED`: The only variables published are those that are poked. These variables are provided on the command line. See Section 2.

Variables subscribed for by the uPokeDB application

- `USER-DEFINED`: Since uPokeDB provides two reports as described in the above Section 5, it subscribes for the same variables it is asked to poke, so it can generate its before-and-after reports.