pAutoPoke: Automated Pokes for Headless Missions

January 2025

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139
`project-pavlab/appdocs/app_pautopoke`

1 Overview

The pAutoPoke application is a tool to enable an automatic poke to the MOOSDB with one or more configured publications. Typically this is in service of conducting headless, auto-tested missions, where there is no user to kick off a mission by hitting a "DEPLOY" button on a command and control GUI, such as apppMarineViewer

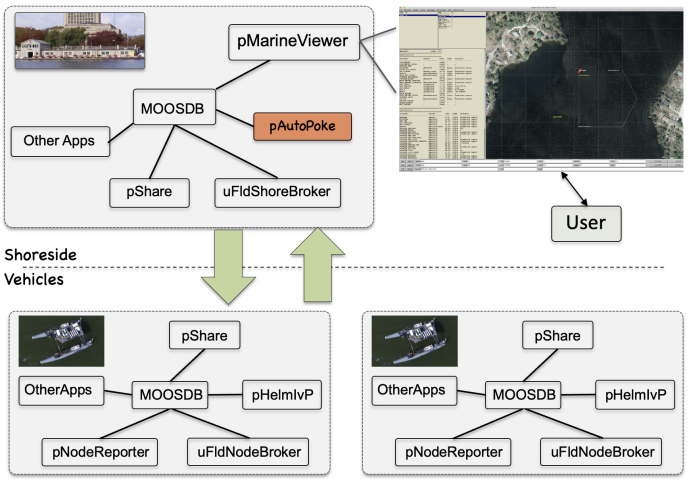

Figure 1: **Typical pAutoPoke Topology:** The flags configured for pAutoPoke are posted immediately after connecting to the MOOSDB. If configured to wait for vehicle connections, the pShoreBroker app will publish the number of connected vehicles in the variable UFSB_NODE_COUNT.

2 Configuration Parameters for pAutoPoke

The pAutoPoke application may be configured with a configuration block within a MOOS mission file, typically with a .moos file suffix. The following parameters are defined for pAutoPoke.

Listing 2.1: Configuration Parameters for pAutoPoke.

flag:	A MOOS variable and value to be published.
required_nodes:	The number of nodes required to be detected, before flags are published. By default this is zero.

An Example MOOS Configuration Block

An example MOOS configuration block may be obtained from the command line with the following:

```
$ pAutoPoke --example or -e
```

Listing 2.2: Example configuration of the pAutoPoke application.

```
 1  ============================================================
 2  pAutoPoke Example MOOS Configuration
 3  ============================================================
 4
 5  ProcessConfig = pAutoPoke
 6  {
 7    AppTick   = 2
 8    CommsTick = 2
 9
10    flag = MOOS_MANUAL_OVERRIDE_ALL=false
11    flag = DEPLOY_ALL=false
12
13    required_nodes = 2
14  }
```

3 Publications and Subscriptions for pAutoPoke

The interface for pAutoPoke, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ pAutoPoke --interface or -i
```

3.1 Variables Published by pAutoPoke

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility.

3.2 Variables Subscribed for by pAutoPoke

The pAutoPoke application subscribes to the following MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- DB_RWSUMMARY: A report generated by the MOOSDB listing, for each app connected to the MOOSDB, the set of variables subscribed for by each app, and the set of variables observed to be published by each app.

Node that pAutoPoke will also subscribe for all variables discovered from the contents of DB_RWSUMMARY publications.

3.3 Command Line Usage of pAutoPoke

The pAutoPoke application is typically launched with pAntler, along with a group of other modules. However, it may be launched separately from the command line. The command line options may be shown by typing:

```
$ pAutoPoke --help or -h
```

Listing 3.3: Command line usage for the pAutoPoke tool.

```
1  Usage: pAutoPoke file.moos [OPTIONS]
2
3  Options:
4    --alias=<ProcessName>
5        Launch pAutoPoke with the given process
6        name rather than pAutoPoke
7    --example, -e
8        Display example MOOS configuration block
9    --help, -h
10       Display this help message.
14   --interface, -i
15       Display MOOS publications and subscriptions.
16   --version,-v
17       Display the release version of pAutoPoke.
18   --web,-w
19       Open browser to: https://oceanai.mit.edu/ivpman/apps/pAutoPoke
20
21 Note: If argv[2] is not of one of the above formats
22       this will be interpreted as a run alias. This
23       is to support pAntler launching conventions.
```

4 Terminal and AppCast Output

Some useful information is published by pAutoPoke to the terminal on every iteration. An example is shown in Listing 4 below. This application is also appcast enabled, meaning its reports are published to the MOOSDB and viewable from any uMAC application or pMarineViewer. The counter on the

end of line 2 is incremented on each iteration of pAutoPoke, and also serves as a heartbeat indicator. The "0/0" also on line 2 indicates there are no configuration or run warnings detected.

Listing 4.4: Example terminal or appcast output for pAutoPoke.

```
 1  ============================================================
 2  pAutoPoke shoreside                                  0/0(44)
 3  ============================================================
 4  Required Nodes:2
 5  Flags:
 6  [0]: MOOS_MANUAL_OVERRIDE_ALL=false
 7  [1]: DEPLOY_ALL=true
 8
 9  Node Count:   2
10  flags_posted: true
```

Lines 4-7 of the output show the user configuration. Lines 9 and 10 show the current state of the app.