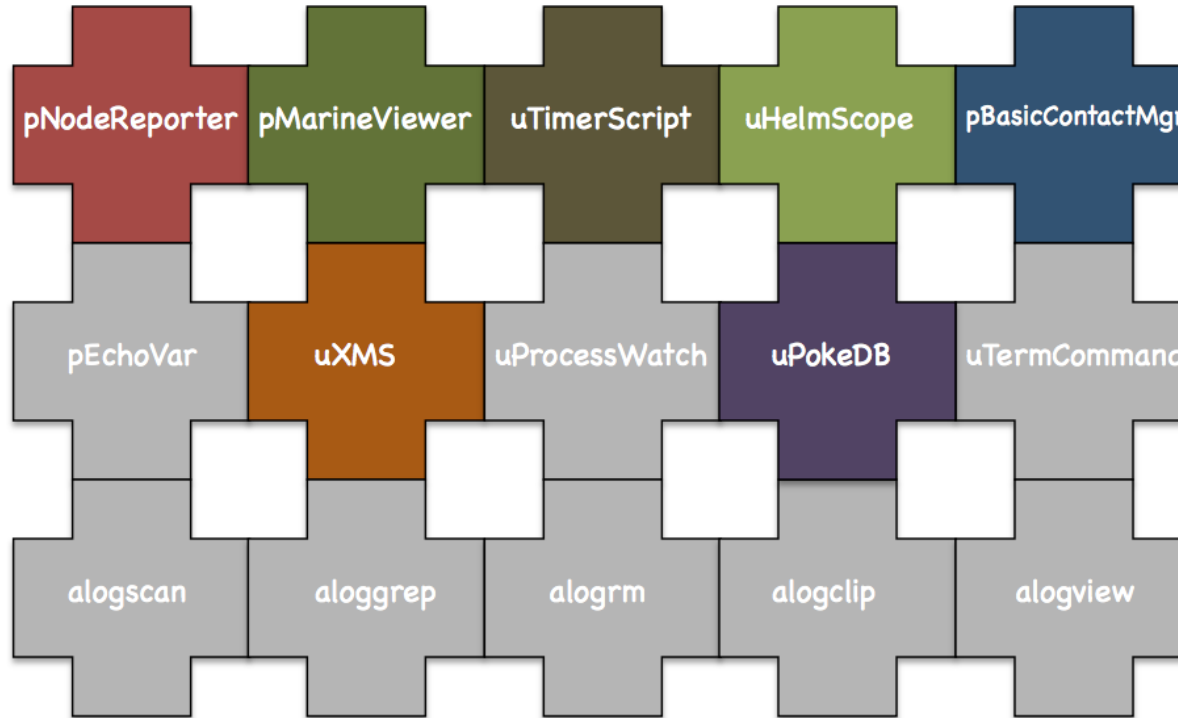




MOOS-IvP Autonomy Tools (A Mini Tutorial)



Michael Benjamin

Code 2501, Center for Advanced Systems Technology
Naval Undersea Warfare Center, Division Newport RI

Approved for public release; Distribution unlimited



Acknowledgments

Collaborators:

This work is the product of a multi-year collaboration between the Center for Advanced System Technologies (CAST), Code 2501, of the Naval Undersea Warfare Center in Newport Rhode Island and the Department of Mechanical Engineering and the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology in Cambridge Massachusetts, and the Oxford University Mobile Robotics Group.

Sponsors:

The IvP Helm autonomy software and the basic research involved in the interval programming model for multi-objective optimization has been developed under support from ONR Code 311 (Program Managers Dr. Don Wagner and Dr. Behzad Kamgar-Parsi). Prior prototype development of IvP concepts benefited from the support of the In-house Laboratory Independent Research (ILIR) program at the Naval Undersea Warfare Center in Newport RI.



Mini-Tutorial Objectives and Structure

Objectives:

Introduce with some depth and examples, the following tools (software applications):

- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string
- uPokeDB - A tool for poking the MOOSDB from the command line.
- uXMS - A tool for focused scoping of the MOOSDB from the console
- uHelmScope - A specialized scope on IvP Helm status and recent history
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB

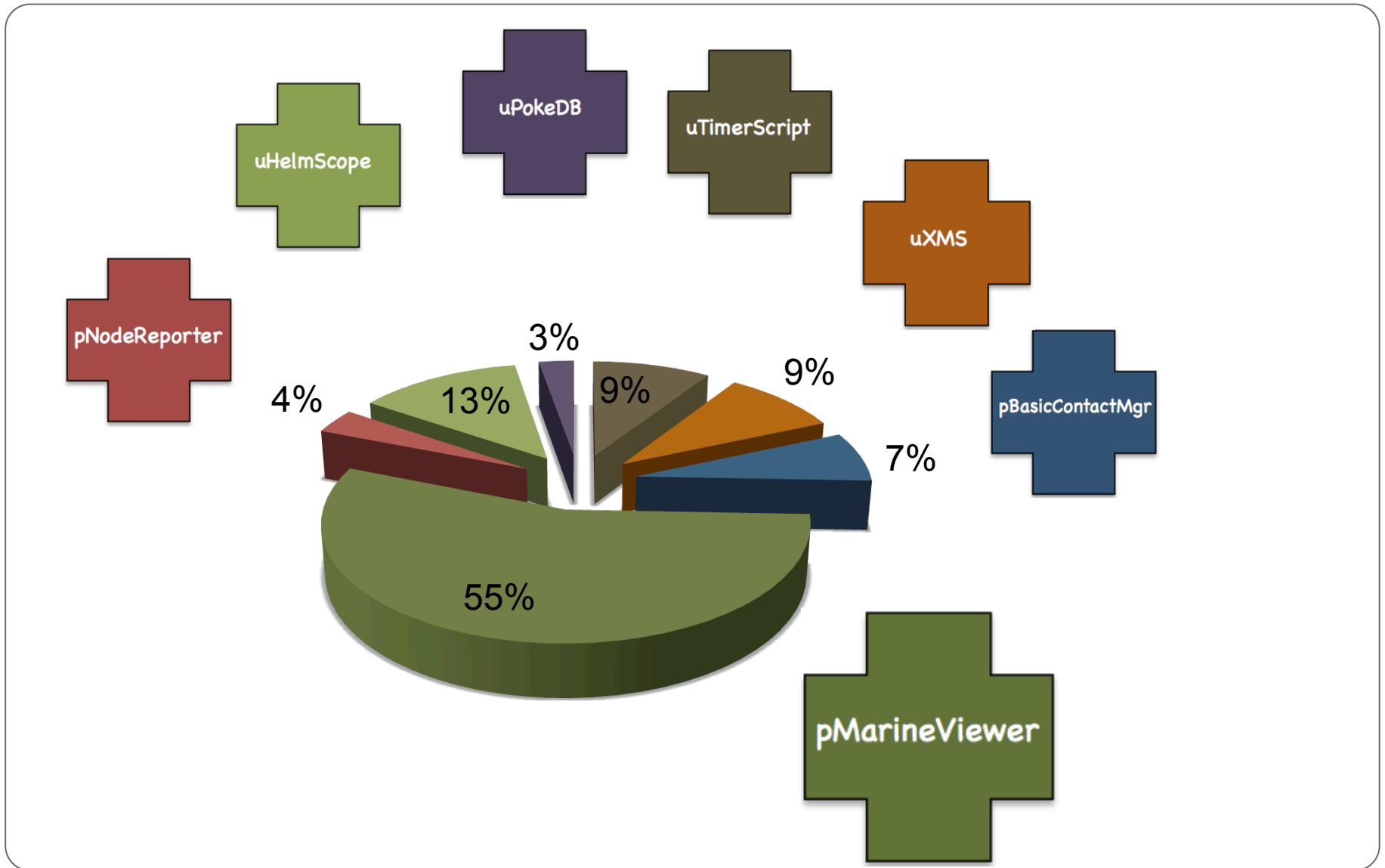
Structure:

The Alpha or Berta example missions will be used to demonstrate each tool.

- These example missions are available along with the MOOS-IvP source code at www.moos-ivp.org
- The example missions can be found under moos-ivp/ivp/missions/.
- It is recommended that tutorial participants download and be able to run these on their laptop computers.

Not All Tools Are Created Equal

Relative Tool Size by Line Count





MOOS-IvP Autonomy Tools

Q: What is an Autonomy Tool?

Ans: A software module/application that supports either (a) the on-board autonomy, (b) pre-mission planning, (c) topside mission monitoring/control, (d) post-mission analysis.

Q: What is the relationship between an Autonomy Tool and the IvP Helm?

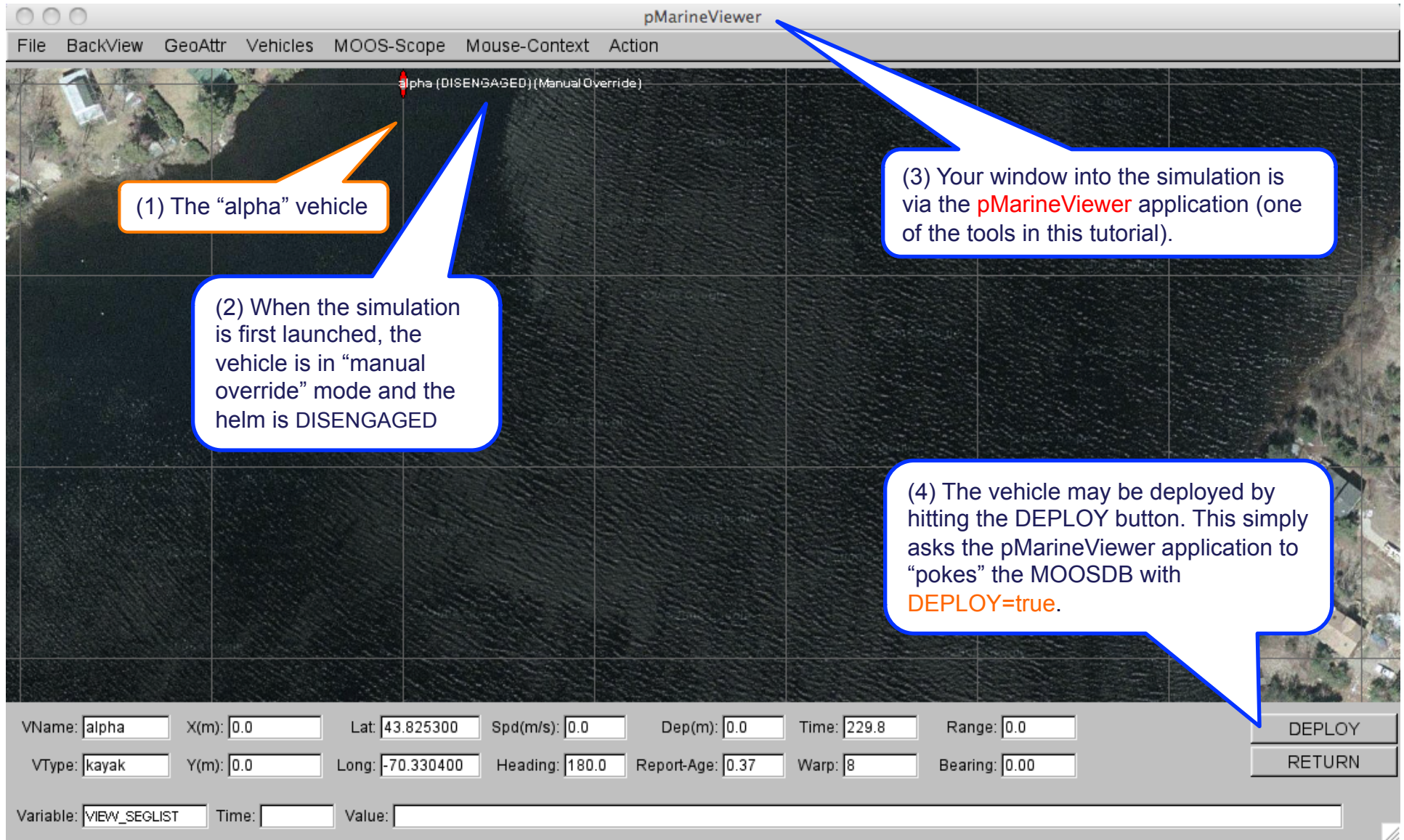
Ans: The IvP Helm is a decision engine that drives the vehicle with heading/speed/depth commands.

- Some tools are specific to the IvP Helm.
- Some tools are general (helm agnostic).
- Some are general but do have some IvP Helm hooks

Q: Where can the Autonomy Tools and documentation be found?

Ans: www.moos-ivp.org

A Walk Through the Alpha Mission (1)



The screenshot shows the pMarineViewer application window. The title bar reads "pMarineViewer". The menu bar includes "File", "BackView", "GeoAttr", "Vehicles", "MOOS-Scope", "Mouse-Context", and "Action". The main display area shows a satellite map with a red icon labeled "alpha (DISENGAGED) (Manual Override)".

Four callout boxes provide instructions:

- (1) The "alpha" vehicle
- (2) When the simulation is first launched, the vehicle is in "manual override" mode and the helm is DISENGAGED
- (3) Your window into the simulation is via the **pMarineViewer** application (one of the tools in this tutorial).
- (4) The vehicle may be deployed by hitting the DEPLOY button. This simply asks the pMarineViewer application to "poke" the MOOSDB with **DEPLOY=true**.

The bottom control panel includes the following fields and buttons:

VName: <input type="text" value="alpha"/>	X(m): <input type="text" value="0.0"/>	Lat: <input type="text" value="43.825300"/>	Spd(m/s): <input type="text" value="0.0"/>	Dep(m): <input type="text" value="0.0"/>	Time: <input type="text" value="229.8"/>	Range: <input type="text" value="0.0"/>	<input type="button" value="DEPLOY"/>
VType: <input type="text" value="kayak"/>	Y(m): <input type="text" value="0.0"/>	Long: <input type="text" value="-70.330400"/>	Heading: <input type="text" value="180.0"/>	Report-Age: <input type="text" value="0.37"/>	Warp: <input type="text" value="8"/>	Bearing: <input type="text" value="0.00"/>	<input type="button" value="RETURN"/>
Variable: <input type="text" value="VIEW_SEGLIST"/>	Time: <input type="text"/>	Value: <input type="text"/>					



A Walk Through the Alpha Mission (2)

pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope Mouse-Context Action

(1) After the vehicle is deployed, by hitting the DEPLOY button, the vehicle begins traversing a set of five waypoints.

Waypoint #1

#5

#4

#3

#2

alpha (ENGAGED)

alpha_deploy_point

alpha_waypoint

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:

A Walk Through the Alpha Mission (3)

pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope Mouse-Context Action

Waypoint #1

#5

#4

#3

#2

alpha (ENGAGED)

(1) After it traverses all five waypoints, it repeats the set once.

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:



A Walk Through the Alpha Mission (4)

pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope Mouse-Context Action

The "return" waypoint

(1) After traversing all five waypoints for the second time, it returns home.

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

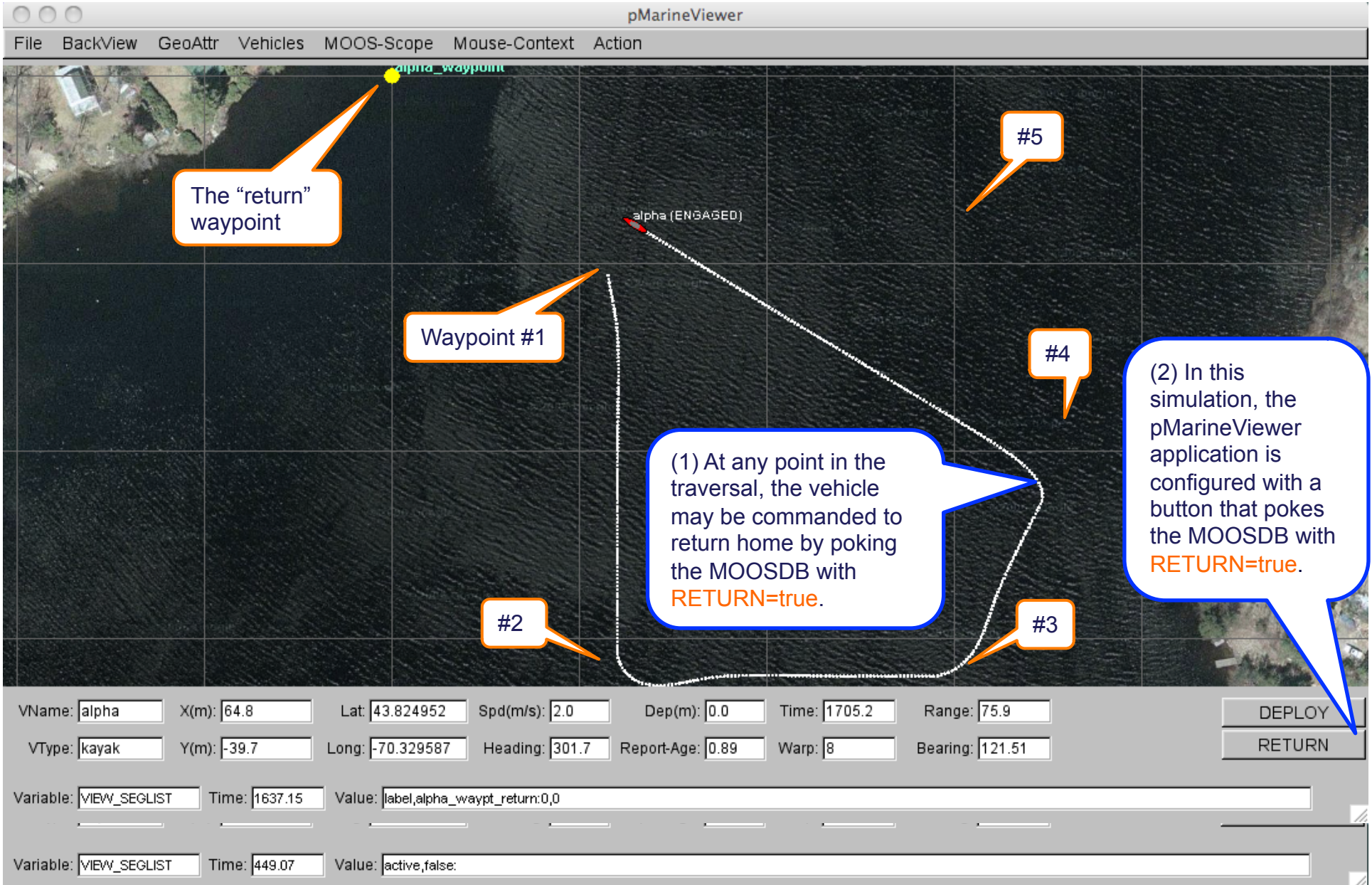
VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:

A Walk Through the Alpha Mission (5)

pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope Mouse-Context Action



The "return" waypoint

Waypoint #1

#2

#3

#4

#5

alpha (ENGAGED)

alpha_waypoint

(1) At any point in the traversal, the vehicle may be commanded to return home by poking the MOOSDB with **RETURN=true**.

(2) In this simulation, the pMarineViewer application is configured with a button that pokes the MOOSDB with **RETURN=true**.

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

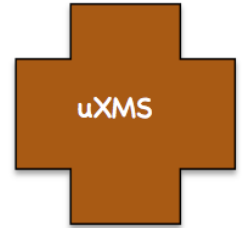
VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:

Variable: Time: Value:



The uXMS Utility: Scoping the MOOSDB from the Console



MOOS Modules:

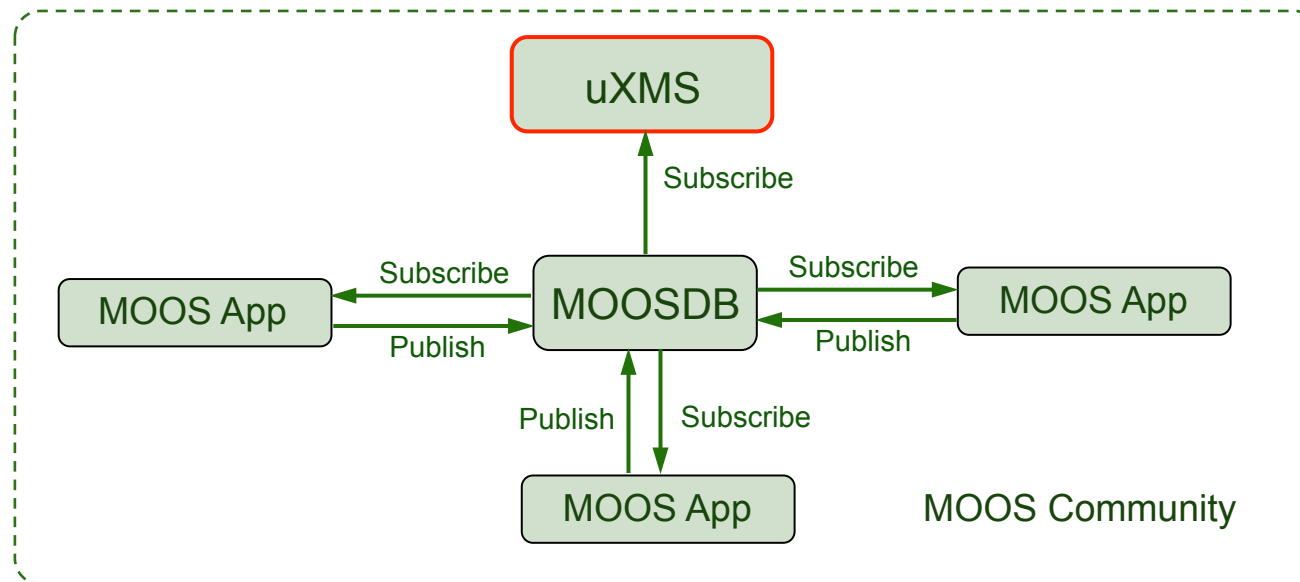
- uXMS - A tool for **focused** scoping of the MOOSDB from the **console**.
- uPokeDB - A tool for poking the MOOSDB from the command line.
- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string.
- uHelmScope - A specialized scope on IvP Helm status and recent history.
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts.
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB.

The uXMS Utility:

Definition of a MOOS Scope and MOOS Community

What is a MOOS *scope*?

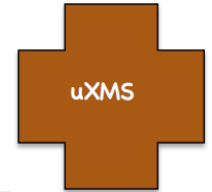
A scope is a tool for monitoring the current state of variables published in a MOOS community.



What is a MOOS *community*?

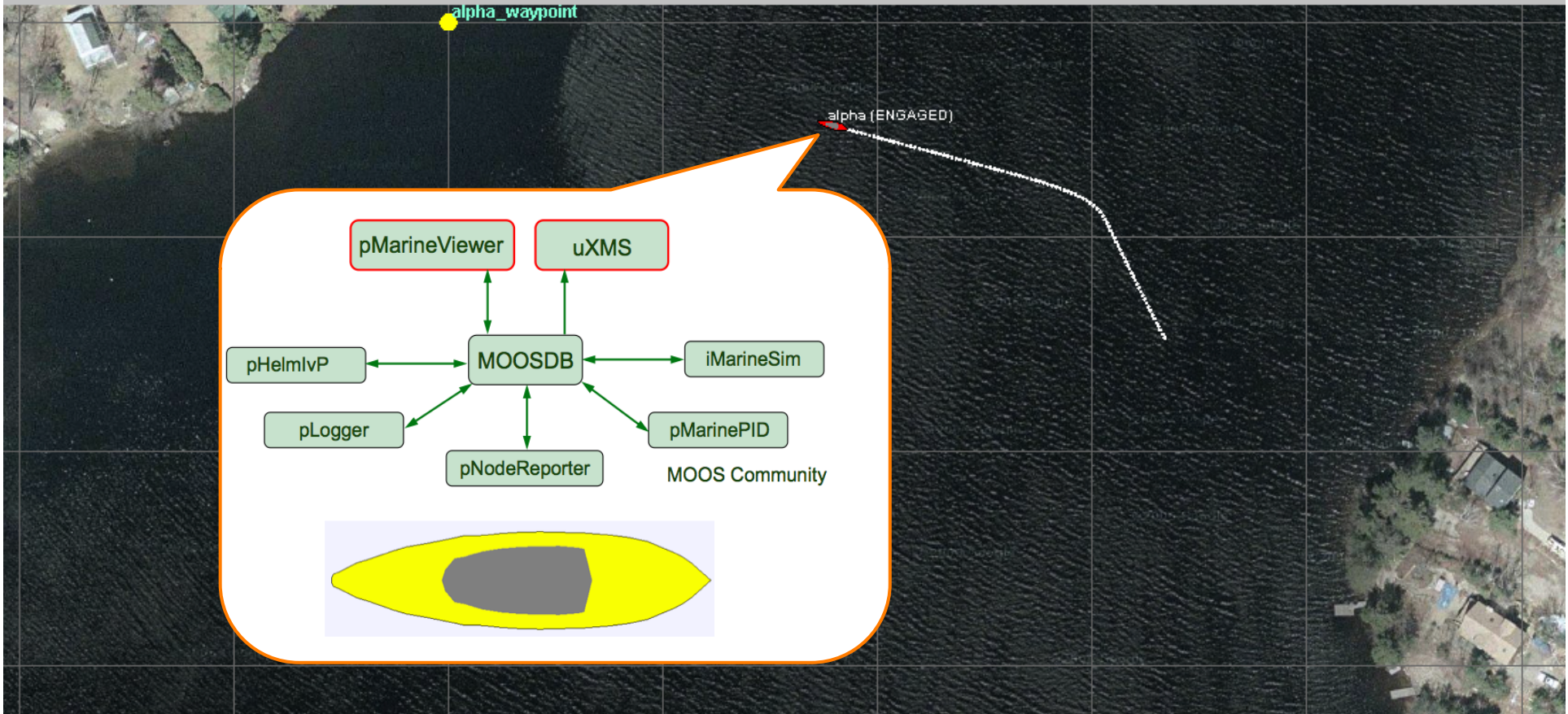
- A collection of MOOS applications connected to the single MOOSDB application.
- Each application interface is defined by what variables it publishes and subscribes to.
- The MOOSDB contains a snapshot of all the current variables – their values and other info.

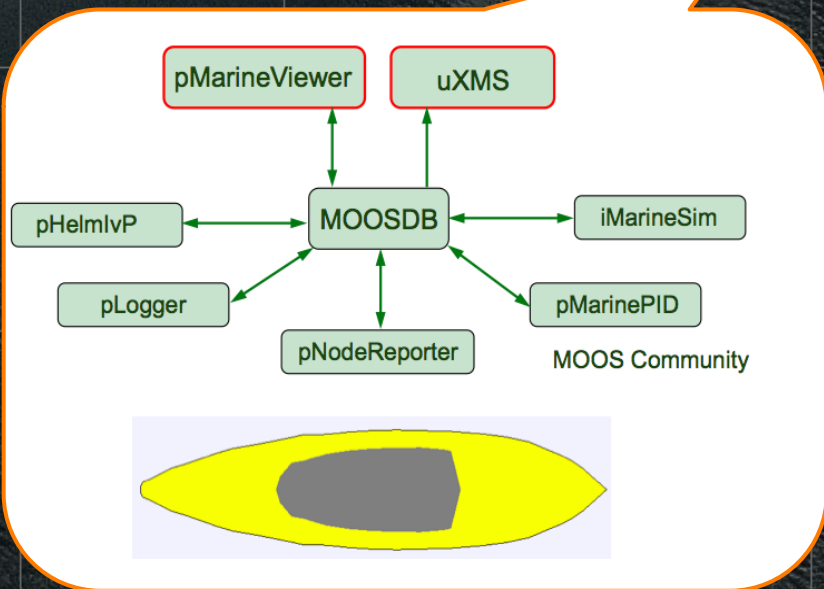
The uXMS Utility: The MOOS Community in the Alpha Mission



pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope Mouse-Context Action

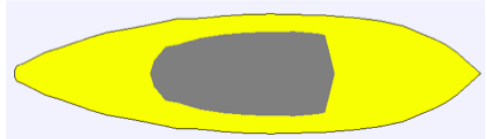




```

graph TD
    pMarineViewer <--> MOOSDB
    uXMS <--> MOOSDB
    pHelmlvP <--> MOOSDB
    iMarineSim <--> MOOSDB
    pLogger <--> MOOSDB
    pNodeReporter <--> MOOSDB
    pMarinePID <--> MOOSDB
    
```

MOOS Community



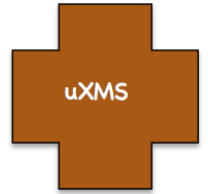
VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:



The uXMS Utility: Launching from the command line



The uXMS utility is launched from the command line:

```
$ uXMS alpha.moos NAV_X NAV_Y NAV_SPEED NAV_HEADING DEPLOY  
MOOS_MANUAL_OVERRIDE DEPLOY IVPHELM_ENGAGED
```

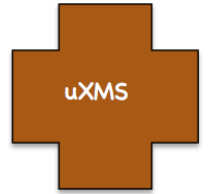
Upon launching, reports are written to the console:

VarName	(S)ource	(T)ime	(C)ommunity	VarValue (MODE = SCOPE:EVENTS)
NAV_X	iMarineSim	1678.99	alpha	0
NAV_Y	iMarineSim	1678.99	alpha	0
NAV_SPEED	iMarineSim	1678.99	alpha	0
NAV_HEADING	iMarineSim	1678.99	alpha	180
DEPLOY	pHelmIvP	5.66	alpha	"false"
MOOS_MANUAL_OVERRIDE	n/a	n/a	n/a	n/a
IVPHELM_ENGAGED	pHelmIvP	1678.03	alpha	"DISENGAGED"

uXMS operates by simply writing a “report” to the console on each iteration.



The uXMS Utility: Specifying the Variables to be Scoped



The variables to be scoped are given on the command line:

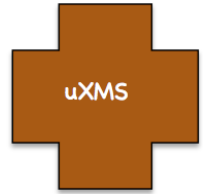
```
$ uXMS alpha.moos NAV_X NAV_Y NAV_SPEED NAV_HEADING DEPLOY  
MOOS_MANUAL_OVERRIDE DEPLOY IVPHELM_ENGAGED
```

Each report dedicates a line to each variable:

VarName	(S)ource	(T)ime	(C)ommunity	VarValue (MODE = SCOPE:EVENTS)
NAV_X	iMarineSim	1678.99	alpha	0
NAV_Y	iMarineSim	1678.99	alpha	0
NAV_SPEED	iMarineSim	1678.99	alpha	0
NAV_HEADING	iMarineSim	1678.99	alpha	180
DEPLOY	pHelmIvP	5.66	alpha	"false"
MOOS_MANUAL_OVERRIDE	n/a	n/a	n/a	n/a
IVPHELM_ENGAGED	pHelmIvP	1678.03	alpha	"DISENGAGED"

The variable values are shown in the fifth and last column:
The variable type (string or double) is indicated by quoting the string values
The value of "n/a" indicates the variable has never been written to.

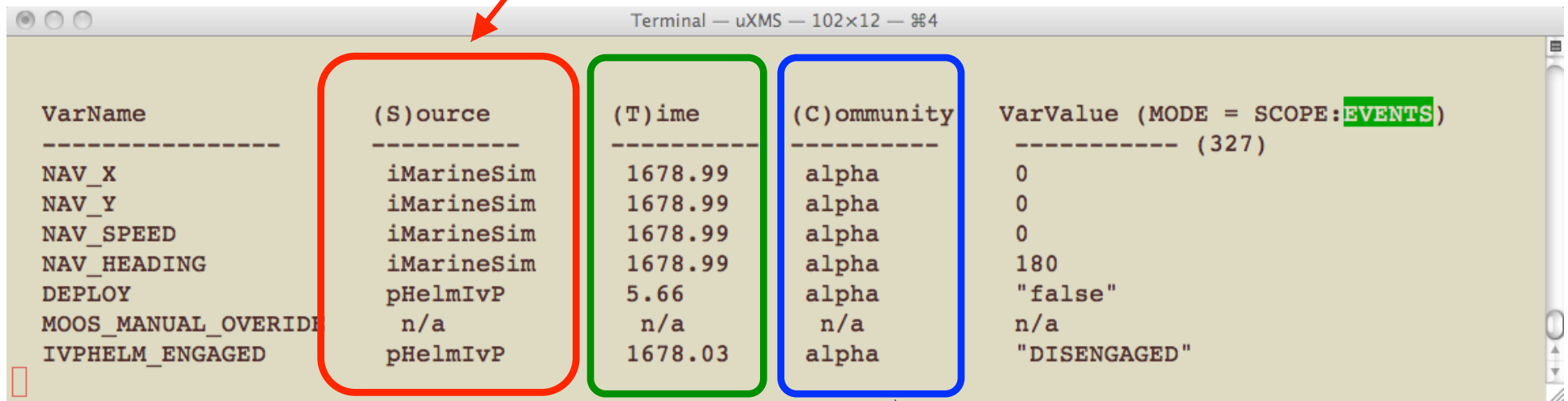
The uXMS Utility: Components of the scope report



The IP address and port number of the MOOSDB is in the .moos file on the command line:

```
$ uXMS alpha.moos NAV_X NAV_Y NAV_SPEED NAV_HEADING DEPLOY
MOOS_MANUAL_OVERRIDE DEPLOY IVPHELM_ENGAGED
```

The 2nd column of each report line shows the *source* of the variable posting:
(The MOOS Application that last published the variable)

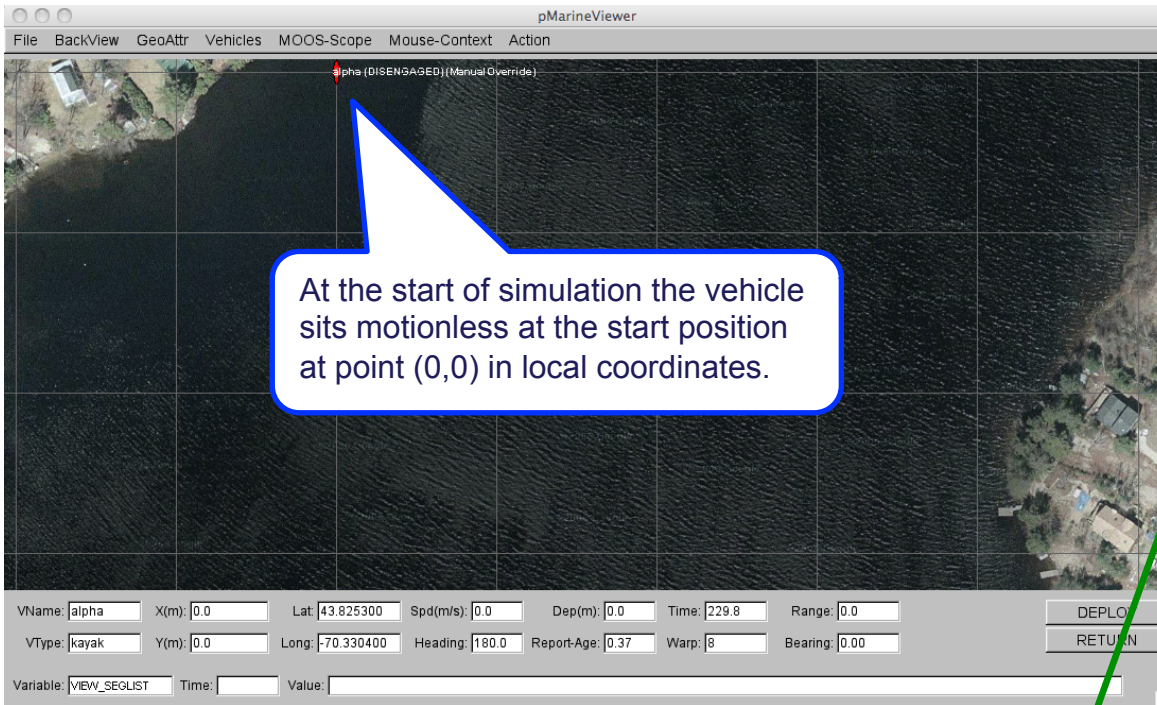
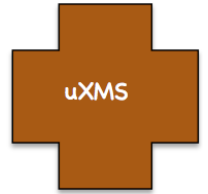


VarName	(S)ource	(T)ime	(C)ommunity	VarValue (MODE = SCOPE:EVENTS)
-----	-----	-----	-----	----- (327)
NAV_X	iMarineSim	1678.99	alpha	0
NAV_Y	iMarineSim	1678.99	alpha	0
NAV_SPEED	iMarineSim	1678.99	alpha	0
NAV_HEADING	iMarineSim	1678.99	alpha	180
DEPLOY	pHelmIvP	5.66	alpha	"false"
MOOS_MANUAL_OVERRIDE	n/a	n/a	n/a	n/a
IVPHELM_ENGAGED	pHelmIvP	1678.03	alpha	"DISENGAGED"

The third column shows the time at which the last posting to the variable was made.

The fourth column shows the name of the MOOS community of the source application.

The uXMS Utility: Scoping on the Alpha Example Mission



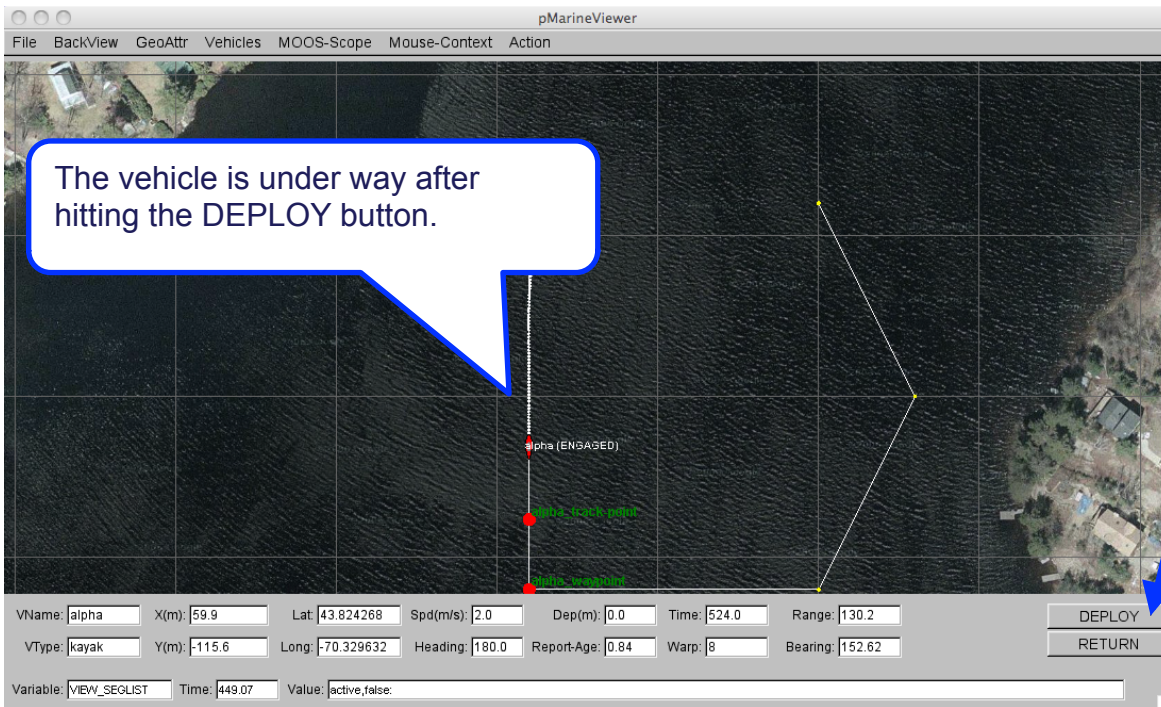
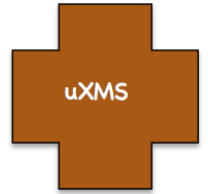
At the outset the vehicle is sitting motionless as shown in the MOOS variables

- NAV_X
- NAV_Y
- NAV_SPEED
- NAV_HEADING

The helm is "DISENGAGED" as evidenced by the MOOS variable IVPHELM_ENGAGED

VarName	(S)ource	(T)ime	(C)ommunity	VarValue (MODE = SCOPE:EVENTS)
NAV_X	iMarineSim	1678.99	alpha	0
NAV_Y	iMarineSim	1678.99	alpha	0
NAV_SPEED	iMarineSim	1678.99	alpha	0
NAV_HEADING	iMarineSim	1678.99	alpha	180
DEPLOY	pHelmIvP	5.66	alpha	"false"
MOOS MANUAL OVERRIDE	n/a	n/a	n/a	n/a
IVPHELM_ENGAGED	pHelmIvP	1678.03	alpha	"DISENGAGED"

The uXMS Utility: Scoping on the Alpha Example Mission



The DEPLOY button is configured to post:

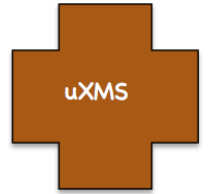
```
DEPLOY=true
MOOS_MANUAL_OVERRIDE=false
```

The postings can be seen in the movement of the vehicle and in the uXMS report:

```
Terminal — uXMS — 102x12 — 84
```

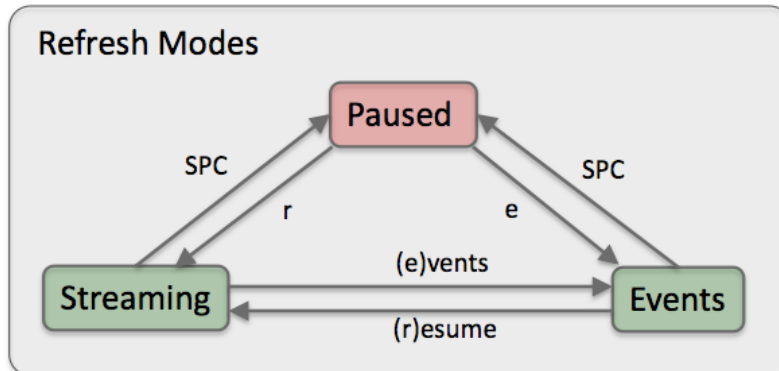
VarName	(S)ource	(T)ime	(C)ommunity	VarValue (MODE = SCOPE:EVENTS)
NAV_X	iMarineSim	1909.37	alpha	159.93792
NAV_Y	iMarineSim	1909.37	alpha	-141.74
NAV_SPEED	iMarineSim	1909.37	alpha	2
NAV_HEADING	iMarineSim	1909.37	alpha	19.36067
DEPLOY	pMarineViewer	1758.2	alpha	"true"
MOOS MANUAL OVERRIDE	pMarineViewer	1758.2	alpha	"false"
IVPHELM_ENGAGED	pHelmIvP	1908.63	alpha	"ENGAGED"

The uXMS Utility: Refresh Modes



The uXMS *refresh mode* determines when a new report is written to the console.

- PAUSED mode: A new report will not be written until user requests it.
- EVENTS mode: A new report is written when a scoped variables changes.
- STREAMING mode: A new report is written on each uXMS iteration.

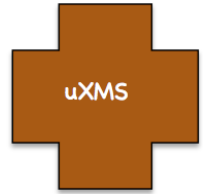


The modes may be switched at the console:

- SPACEBAR – pauses the scope and requests a single new report.
- ‘e’ or ‘E’ – moves the scope into Events mode.
- ‘r’ or ‘R’ – moves the scope into Streaming mode. (‘s’ is reserved for something else).

- The PAUSED mode is a key feature of uXMS – it minimizes communications bandwidth.
- The default mode is the EVENTS mode.
- The mode at launch time may be change by specifying ‘--mode=paused’ on command line.

The uXMS Utility: Refresh Mode Indicator



The uXMS *refresh mode* is indicated in the top right-hand corner of each report:

```
Terminal — uXMS — 105x11 — %2
VarName      (S)ource      (T)ime      (C)ommunity  VarValue (MODE = SCOPE:STREAMING)
-----
NAV_X        iMarineSim    425.51     alpha       178.07585
NAV_Y        iMarineSim    425.51     alpha       -94.67142
NAV_SPEED    iMarineSim    425.51     alpha       2
NAV_HEADING  iMarineSim    425.51     alpha       346.37853
DEPLOY       pMarineViewer 248.56     alpha       "true"
MOOS_MANUAL_OVERRIDE pMarineViewer 248.56     alpha       "false"
IVPHELM_ENGAGED pHelmIvP     425.38     alpha       "ENGAGED"
```

STREAMING

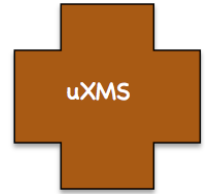
```
Terminal — uXMS — 105x11 — %2
VarName      (S)ource      (T)ime      (C)ommunity  VarValue (MODE = SCOPE:PAUSED)
-----
NAV_X        iMarineSim    436.76     alpha       167.50418
NAV_Y        iMarineSim    436.76     alpha       -74.80995
NAV_SPEED    iMarineSim    436.76     alpha       2
NAV_HEADING  iMarineSim    436.76     alpha       331.11722
DEPLOY       pMarineViewer 248.56     alpha       "true"
MOOS_MANUAL_OVERRIDE pMarineViewer 248.56     alpha       "false"
IVPHELM_ENGAGED pHelmIvP     436.4      alpha       "ENGAGED"
```

PAUSED

EVENTS

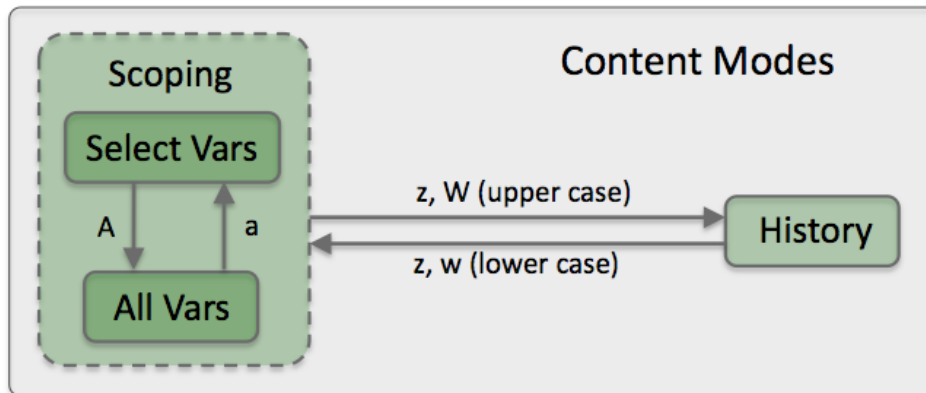
```
Terminal — uXMS — 105x11 — %2
VarName      (S)ource      (T)ime      (C)ommunity  VarValue (MODE = SCOPE:EVENTS)
-----
NAV_X        iMarineSim    488.39     alpha       84.91045
NAV_Y        iMarineSim    488.39     alpha       -40.13027
NAV_SPEED    iMarineSim    488.39     alpha       2
NAV_HEADING  iMarineSim    488.39     alpha       270.01851
DEPLOY       pMarineViewer 248.56     alpha       "true"
MOOS_MANUAL_OVERRIDE pMarineViewer 248.56     alpha       "false"
IVPHELM_ENGAGED pHelmIvP     487.52     alpha       "ENGAGED"
```


The uXMS Utility: Content Modes



The uXMS *content mode* determines what is written in the reports to the console.

- SCOPING mode: A report contains a snapshot of variables in the scope list.
- HISTORY mode: A report contains the recent history of given variable.



The modes may be switched at the console:

- 'W' always puts uXMS into History mode.
- 'w' always puts uXMS into Scoping mode.
- 'z' or 'Z' – toggles uXMS between Scoping and History mode.

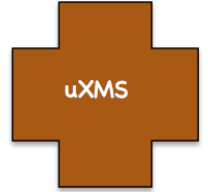
The SCOPING has two sub-modes:

- The "SelectVars" mode reports only on variables in the configured scope list.
- The "AllVars" mode reports on ALL variables known to the MOOSDB.

The modes may be switched at the console:

- 'A' always puts uXMS into SelectVars Scoping mode.
- 'a' always puts uXMS into AllVars Scoping mode.

The uXMS Utility: The “History” Content Mode

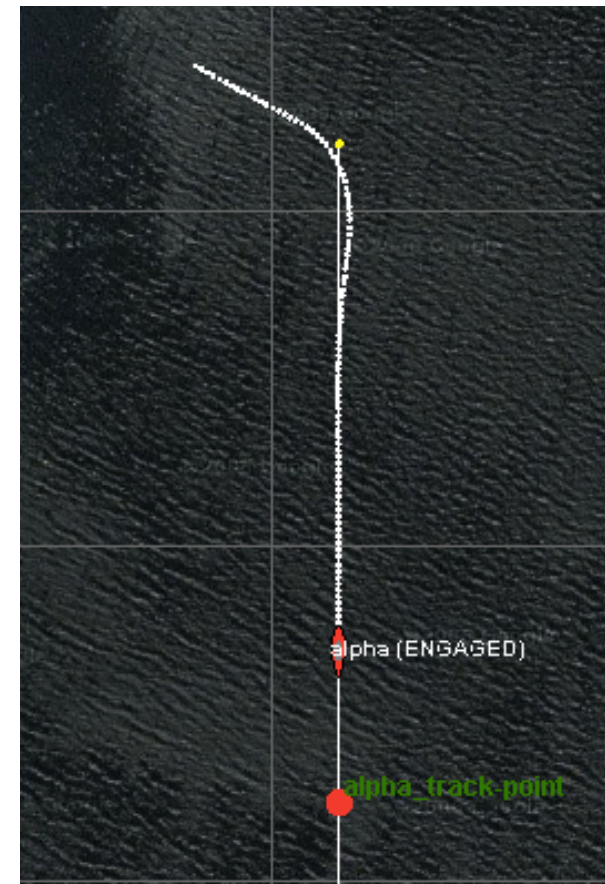


The uXMS *History* content mode shows the recent history of a specified MOOS variable.

- The history is limited to 20 lines (configurable)
- The refresh mode may also be set to EVENTS, PAUSED or STREAMING.

```
Terminal — uXMS — 91x24 — %6

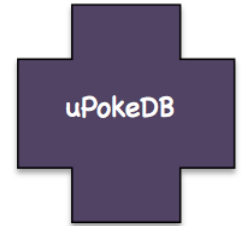
-----
VarName      (S)ource    (T)ime      VarValue (MODE = HISTORY:EVENTS)
-----
DESIRED_HEADING  pHelmIvP    50.82      (1) 183
DESIRED_HEADING  pHelmIvP    51.07      (1) 184
DESIRED_HEADING  pHelmIvP    51.32      (1) 186
DESIRED_HEADING  pHelmIvP    51.82      (2) 188
DESIRED_HEADING  pHelmIvP    52.32      (2) 189
DESIRED_HEADING  pHelmIvP    52.82      (2) 190
DESIRED_HEADING  pHelmIvP    54.82      (8) 191
DESIRED_HEADING  pHelmIvP    55.83      (4) 190
DESIRED_HEADING  pHelmIvP    56.33      (2) 189
DESIRED_HEADING  pHelmIvP    57.33      (4) 188
DESIRED_HEADING  pHelmIvP    57.58      (1) 187
DESIRED_HEADING  pHelmIvP    57.83      (1) 186
DESIRED_HEADING  pHelmIvP    58.08      (1) 185
DESIRED_HEADING  pHelmIvP    58.58      (2) 184
DESIRED_HEADING  pHelmIvP    59.08      (2) 183
DESIRED_HEADING  pHelmIvP    59.83      (3) 182
DESIRED_HEADING  pHelmIvP    61.33      (6) 181
DESIRED_HEADING  pHelmIvP    67.08      (23) 180
DESIRED_HEADING  pHelmIvP    70.32      (13) 179
DESIRED_HEADING  pHelmIvP    84.85      (58) 180
-----
```



Successive duplicate entries are condensed into a single line with the number of duplicates indicated in parentheses.



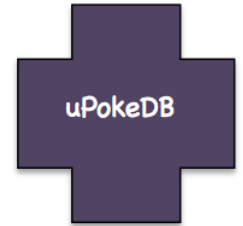
The uPokeDB Utility: Poking the MOOSDB from the Console



MOOS Modules:

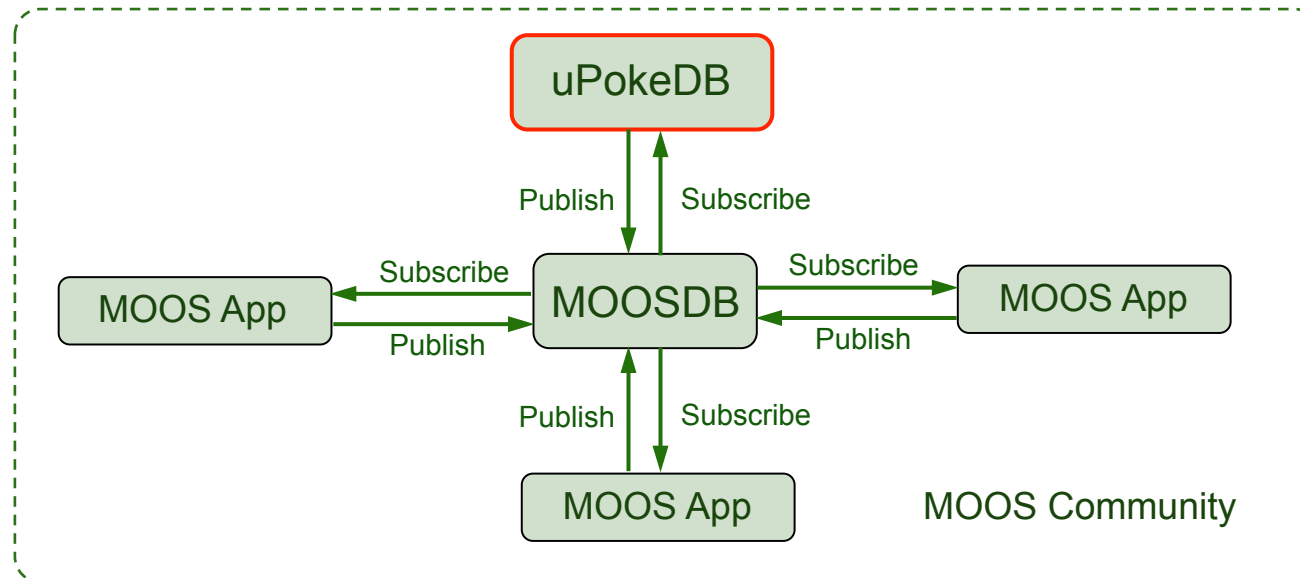
- uXMS - A tool for focused scoping of the MOOSDB from the console.
- uPokeDB - A tool for poking the MOOSDB from the command line.
- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string.
- uHelmScope - A specialized scope on IvP Helm status and recent history.
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts.
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB.

The uPokeDB Utility: Definition of a MOOS Poke



What is a MOOS *poke*?

A poke is the publication of a MOOS variable-value pair to a given MOOSDB. The “poke” implies that publication is “one-time” event.



- uPokeDB primarily *publishes* to the MOOSDB (the poke).
- uPokeDB also *subscribes* to the MOOSDB for mail on the variable it is poking – to show the user the variable value prior to the poke, and confirm the variable value after the poke.

The uPokeDB Utility: Launching the Utility and Understanding the Output

The uPokeDB utility is launched from the command line:

```
$ uPokeDB alpha.moos DEPLOY=true MOOS_MANUAL_OVERRIDE=false
```

```
Terminal -- tcsh -- 76x35 -- %6
$ uPokeDB alpha.moos DEPLOY=true MOOS_MANUAL_OVERRIDE=false
Mission File was provided: alpha.moos
*****
*                               *
*   This is MOOS Client         *
*   c. P Newman 2001           *
*                               *
*****
-----MOOS CONNECT-----
contacting a MOOS server localhost:9000 - try 00001
Contact Made
Handshaking as "uPokeDB"
Handshaking Complete
Invoking User OnConnect() callback...ok
-----

uPokeDB is Running:
  AppTick   @ 5.0 Hz
  CommsTick @ 5 Hz

PRIOR to Poking the MOOSDB
VarName      (S)ource  (T)ime  VarValue
-----
DEPLOY       pHelmIvP  0.78   "false"
MOOS_MANUAL_OVERRIDE

AFTER Poking the MOOSDB
VarName      (S)ource  (T)ime  VarValue
-----
DEPLOY       uPokeDB   7.23   "true"
MOOS MANUAL OVERRIDE  uPokeDB   7.23   "false"

$
```

(1) Upon launching uPokeDB will confirm connection to the MOOSDB.

(2) It will write to stdout, the current values of the variables being poked.

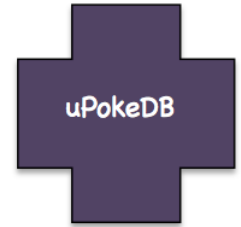
(3) It will confirm the new values of the variables after the poke – as seen by the mail received by the MOOSDB.

(4) It will then quit.



The uPokeDB Utility:

Other ways of Poking the MOOSDB



- A “Poke” is just publication to the MOOSDB, no different than the publications that occur when a MOOS application publishes/writes/posts to the MOOSDB.
 - A publication is only “poke” because it is regarded as being outside the “normal” set of variables published by that particular application.
-

Some other utilities and methods for Poking The MOOSDB:

(1) iRemote:

- It may be configured to associate a poke with any unmapped key.

(2) uTermCommand:

- A utility for configuring user-defined pokes (variable-value pairs) with a unique key word. uTermCommand then allows the user to type in the key word and trigger the poke. The key word may trigger more than one poke if desired.
- The uTermCommand utility is in the moos-ivp tree.

(3) pMarineViewer:

- On-screen buttons may be configured to trigger one or more user-defined pokes.
- An “Action” pull-down menu may be configure to associate a pull-down menu item with one or more pokes.



The pMarineViewer Utility: A GUI for Mission Control



MOOS Modules:

- uXMS - A tool for focused scoping of the MOOSDB from the console.
- uPokeDB - A tool for poking the MOOSDB from the command line.
- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string.
- uHelmScope - A specialized scope on IvP Helm status and recent history.
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts.
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB.



The pMarineViewer Utility: What it is, and is not



What is pMarineViewer?

- A run-time tool for rendering one or more vehicles during operation or simulation.
- Rendering is possible on a geographical map, given a map image and coordinates.
- Geometric objects, e.g., a set of waypoints or polygon, may also be rendered.
- It may be used for command-and-control by configuring pokes to the local MOOSDB.

What pMarineViewer is NOT:

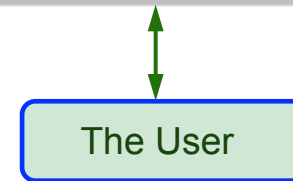
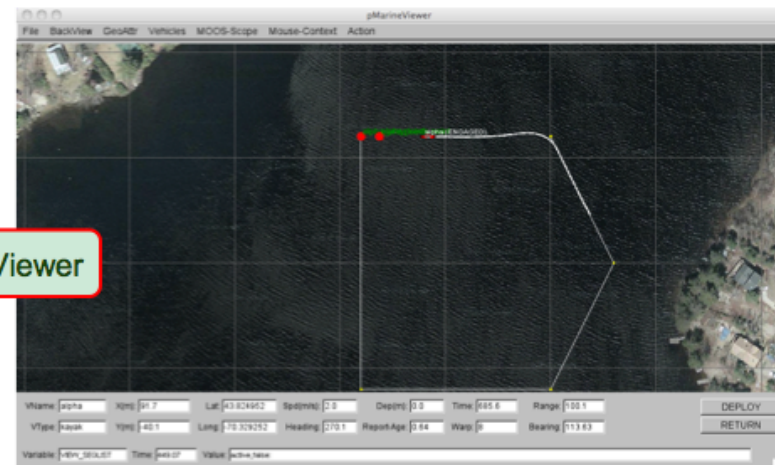
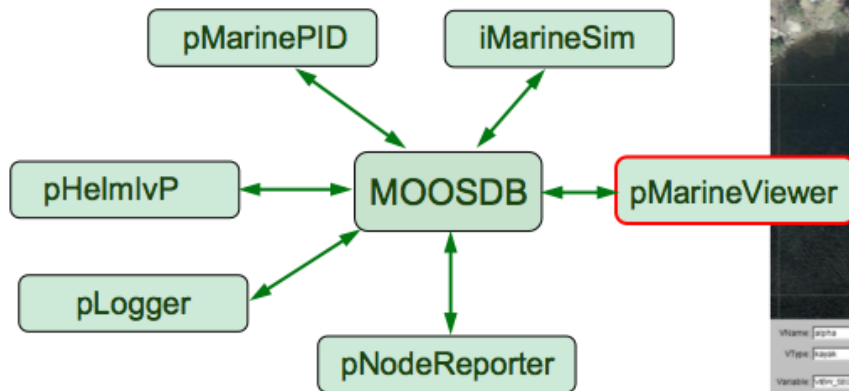
- It is not a mission-planning tool.
- It is not a post-mission analysis tool (unless using the uPlayback utility).
- It is not capable of “pausing” or moving back in time.
- It does not have any communications capability to other MOOS communities, local or remote.

The pMarineViewer Utility: One Simple MOOS Community Topology



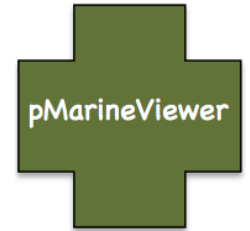
A simple topology, used in the Alpha example mission:

- pMarineViewer connects to the same MOOSDB (MOOS Community) running the vehicle.



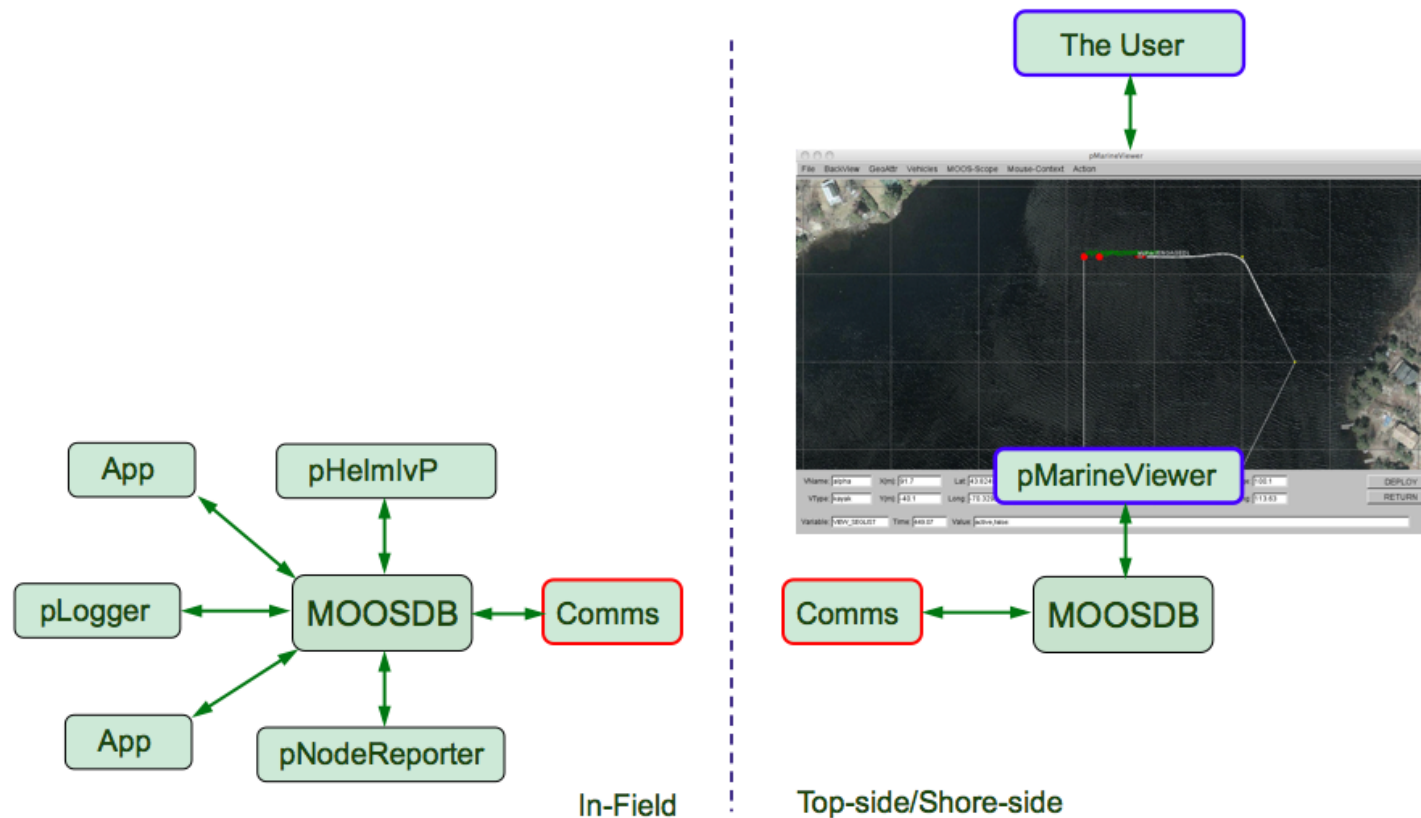
- The User interacts with the GUI to alter the rendering perspective and poke commands to the MOOSDB.
- pNodeReporter publishes NODE_REPORT postings, read by pMarineViewer to update vehicle positions.
- pHelmlvP publishes geometric artifacts like waypoints, read by pMarineViewer and rendered.

The pMarineViewer Utility: A More Extendable MOOS Community Topology



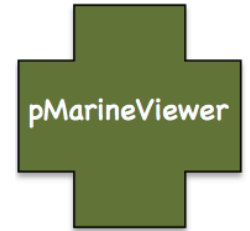
A simple topology, used in the Alpha example mission:

- pMarineViewer runs in its own dedicated MOOS Community – typically on a different machine.



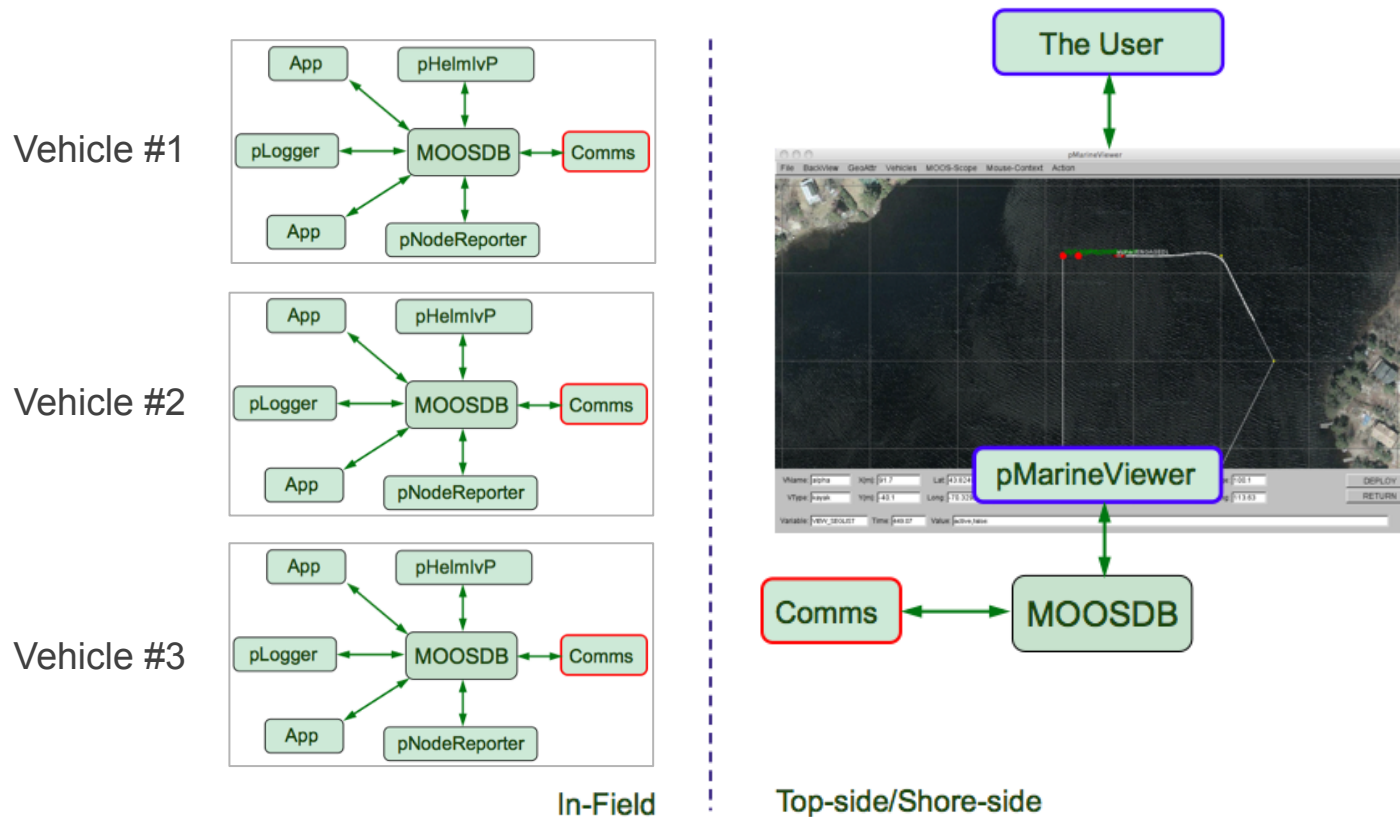
- The Comms connection may be Acomms, Wifi, Iridium, or just may be running on the same machine.
- There may be an arbitrary number of vehicles connected to the pMarineViewer Community.

The pMarineViewer Utility: A More Extendable MOOS Community Topology



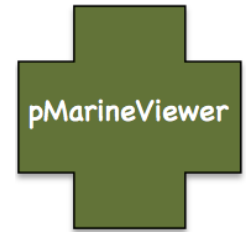
A simple topology, used in the Alpha example mission:

- pMarineViewer runs in its own dedicated MOOS Community – typically on a different machine.



- The Comms connection may be Acomms, Wifi, Iridium, or just may be running on the same machine.
- There may be an arbitrary number of vehicles connected to the pMarineViewer Community.

The pMarineViewer Utility: The BackView Pull-Down Menu



The screenshot shows the pMarineViewer application window with the 'BackView' menu open. The menu items and their keyboard shortcuts are as follows:

Zoom In	I
Zoom Out	O
Zoom Reset	Ctrl+Z
Pan Up	Up
Pan Down	Down
Pan Left	Left
Pan Right	Right
Pan Up (slow)	Option+Up
Pan Down (slow)	Option+Down
Pan Left (slow)	Option+Left
Pan Right (slow)	Option+Right
Pan Up (v. slow)	Ctrl+Up
Pan Down (v. slow)	Ctrl+Down
Pan Left (v. slow)	Ctrl+Left
Pan Right (v. slow)	Ctrl+Right
tiff_view toggle	B
tiff_type toggle	-
back_shade lighter	Ctrl+B
back_shade darker	Option+B
hash_view toggle	H
hash_shade lighter	Ctrl+H
hash_shade darker	Option+H
<input type="radio"/> hash_delta=10	Option+1
<input type="radio"/> hash_delta=50	Option+2
<input type="radio"/> hash_delta=100	Option+3
<input type="radio"/> hash_delta=200	Option+4
<input type="radio"/> hash_delta=500	Option+5
<input type="radio"/> hash_delta=1000	Option+6
<input type="radio"/> Hash Auto	Option+7

Callouts in the image point to the following menu items:

- Zoom Control (points to Zoom In, Zoom Out, Zoom Reset)
- Pan Control (regular) (points to Pan Up, Pan Down, Pan Left, Pan Right)
- Pan Control (Slow) (points to Pan Up (slow), Pan Down (slow), Pan Left (slow), Pan Right (slow))
- Pan Control (Very Slow) (points to Pan Up (v. slow), Pan Down (v. slow), Pan Left (v. slow), Pan Right (v. slow))
- Turn off/on the background image (points to tiff_view toggle)
- Toggle the hash marks and adjust shading (points to hash_view toggle, hash_shade lighter, hash_shade darker)
- Control the hash mark spacing (points to the hash_delta radio button options)
- Hash Marks (points to the hash marks on the map)

The bottom of the window contains control fields for Time (524.0), Range (130.2), Warp (8), and Bearing (152.62), along with DEPLOY and RETURN buttons.

The pMarineViewer Utility: The GeoAttributes Pull-Down Menu



The screenshot shows the pMarineViewer application interface. The 'GeoAttr' menu is open, listing various geographic features and their corresponding edit and toggle options. Annotations in blue boxes point to these menu items, explaining their functions. Orange boxes and arrows highlight specific features on the map: 'XYSegList' (a yellow point) and 'XYPoints' (red points labeled 'alpha (ENGAGED)', 'alpha - back point', and 'alpha - forward point'). The bottom of the screen displays a data entry form for the selected object 'alpha'.

GeoAttribute	Toggle Key	Function
Polygons - Edit		Adjust default XYPolygon attributes
Polygons - Toggle	P	
SegLists - Edit		Adjust default XYSegList attributes
SegLists - Toggle	S	
Points - Edit		Adjust default XYPoint attributes
Points - Toggle	J	
XYGrids - Edit		Adjust default XYGrid attributes
XYGrids - Toggle	G	
Datum - Edit		Toggle the rendering of the Datum
Datum - Toggle	D	
Markers - Edit		Adjust default Markers attributes
Markers - Toggle	M	
OpArea - Edit		Adjust default OpArea attributes
OpArea - Toggle	U	
DropPoints - Edit		
DropPoints - Toggle	R	

VName:	alpha	X(m):	59.9	Lat:	43.824268	Spd(m/s):	2.0	Dep(m):	0.0	Time:	524.0	Range:	130.2	DEPLOY
VType:	kayak	Y(m):	-115.6	Long:	-70.329632	Heading:	180.0	Report-Age:	0.84	Warp:	8	Bearing:	152.62	RETURN
Variable:	VIEW_SEGLIST	Time:	449.07	Value:	active,false:									

The pMarineViewer Utility: The GeoAttributes Pull-Down Menu



The screenshot shows the pMarineViewer application window with the 'GeoAttr' menu open. The menu items are:

- Polygons - Edit
- Polygons - Toggle P
- SegLists - Edit
- SegLists - Toggle S
- Points - Edit
- Points - Toggle J
- XYGrids - Edit
- XYGrids - Toggle G
- Datum - Edit
- Datum - Toggle D
- Markers - Edit
- Markers - Toggle M
- OpArea - Edit
- OpArea - Toggle U
- DropPoints - Edit
- DropPoints - Toggle R

Annotations and callouts:

- A blue callout box points to the 'Polygons - Toggle P', 'SegLists - Toggle S', 'Points - Toggle J', and 'XYGrids - Toggle G' items, stating: "Toggle or adjust default attributes for the XYPolygon, XYSegList, XYPoint, and XYGrid objects."
- A blue callout box points to the 'Datum - Toggle D' item, stating: "Toggle the rendering of the Datum"
- A blue callout box points to the 'Markers - Toggle M', 'OpArea - Toggle U', and 'DropPoints - Toggle R' items, stating: "Toggle or adjust default attributes for Markers, OpArea and DropPoint objects."
- An orange callout box labeled 'XYSegList' points to a white line segment on the map.
- An orange callout box labeled 'XYPoints' points to two red circular markers on the map.

The application status bar at the bottom displays the following data:

VName: alpha	X(m): 59.9	Lat: 43.824208	Spd(m/s): 2.0	Dep(m): 0.0	Time: 524.0	Range: 130.2	DEPLOY	
VType: kayak	Y(m): -115.6	Long: -70.329632	Heading: 180.0	Report-Age: 0.84	Warp: 8	Bearing: 152.62	RETURN	
Variable: VIEW_SEGLIST	Time: 449.07	Value: active,false:						



The pMarineViewer Utility: The Vehicles Pull-Down Menu



The screenshot shows the pMarineViewer application window with the 'Vehicles' menu open. The menu items are: Vehicles-Toggle (Ctrl+V), Cycle Focus (V), center_view = vehicle_average (C), center_view = vehicle_active (Ctrl+C), VehicleSize, VehicleNames, Trails, BearingLines, ActiveColor, and InactiveColor. Annotations with blue boxes and lines point to these items and the map area. The map shows a satellite view of a body of water with a grid overlay and three vehicle icons: 'alpha (ENGAGED)' (red), 'alpha_black-pool' (green), and 'alpha_wagtail' (green). A white line connects the vehicles, and a yellow dot is on the map. The bottom of the window has a data panel with fields for VName, X(m), Lat, Spd(m/s), Dep(m), Time, Range, VType, Y(m), Long, Heading, Report-Age, Warp, Bearing, and buttons for DEPLOY and RETURN. A variable field shows 'VIEW_SEGLIST' with a time of 449.07 and a value of 'active,false:'.

Vehicles Pull-Down Menu:

- Vehicles-Toggle (Ctrl+V)
- Cycle Focus (V)
- center_view = vehicle_average (C)
- center_view = vehicle_active (Ctrl+C)
- VehicleSize
- VehicleNames
- Trails
- BearingLines
- ActiveColor
- InactiveColor

Annotations:

- Toggle off/on the rendering of vehicles (points to Vehicles-Toggle)
- Adjust which vehicle is "active". This vehicle has its info displayed in the fields at the bottom of the viewer (points to center_view = vehicle_active)
- Pan the viewer to the vehicle (points to the map area)
- Toggle off/on the rendering vehicle trails and adjust length (points to Trails)
- Change default color of vehicles (points to ActiveColor)

Data Panel:

VName:	alpha	X(m):	59.9	Lat:	43.824268	Spd(m/s):	2.0	Dep(m):	0.0	Time:	524.0	Range:	130.2	DEPLOY
VType:	kayak	Y(m):	-115.6	Long:	-70.329632	Heading:	180.0	Report-Age:	0.84	Warp:	8	Bearing:	152.62	RETURN
Variable:	VIEW_SEGLIST	Time:	449.07	Value:	active,false:									

The pMarineViewer Utility: The MOOS-Scope Pull-Down Menu



The screenshot shows the pMarineViewer application window with the MOOS-Scope menu open. The menu items are:

- Add Variable A
- Toggle-Previous-Scope /
- Cycle-Scope-Variables Ctrl+V
- VIEW_SEGLIST
- VIEW_POINT
- VIEW_POLYGON
- NAV_X
- NAV_Y
- MVIEWER_LCLICK
- MVIEWER_RCLICK

Annotations on the screenshot:

- Add a MOOS variable to be Scoped in the Scope field.** (Points to the 'Add Variable' menu item)
- The list of all variables identified for scoping.** (Points to the list of menu items)
- Cycle between all variables identified for scoping.** (Points to the 'Cycle-Scope-Variables' menu item)
- Toggle between the two variables most recently chosen for scoping** (Points to the 'Toggle-Previous-Scope' menu item)

At the bottom of the window, there are several data fields:

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:



The pNodeReporter Utility: Summarizing a Node's Status



MOOS Modules:

- uXMS - A tool for focused scoping of the MOOSDB from the console.
- uPokeDB - A tool for poking the MOOSDB from the command line.
- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string.
- uHelmScope - ~~A specialized scope on IvP Helm status and recent history.~~
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts.
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB.



The pNodeReporter Utility: What it is, and is not



What is pNodeReporter?

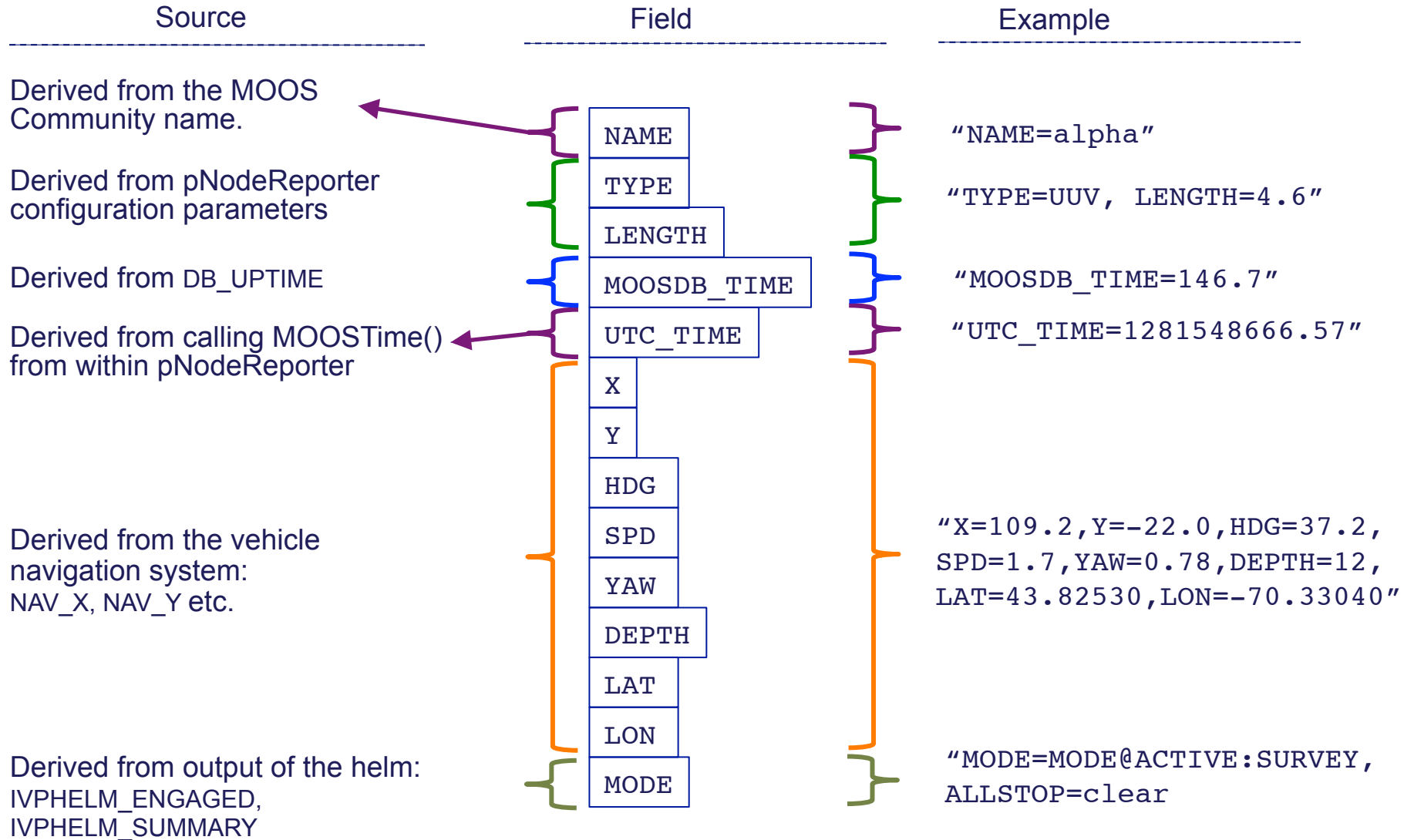
- A run-time tool for gathering information about ownship and summarizing in a single MOOS variable.
- The single MOOS variable is NODE_REPORT_LOCAL.
- It reports information on the platform position and trajectory.
- It reports information on the platform type, and length.
- It reports certain key information regarding the state of the IvP Helm.
- It can be viewed as a loose proxy for an AIS (Automatic Information System) report.

What pNodeReporter is NOT:

- It does not handle communications between platforms.
- It does not handle incoming reports from other platforms.



The pNodeReporter Utility: Basic Functions





The pNodeReporter Utility: Alpha Example Mission



In the Alpha Example Mission:

- Launch the mission
- Run uXMS with:

```
$ uXMS -history=NODE_REPORT_LOCAL
```

- Note the successive values of NODE_REPORT_LOCAL reported.

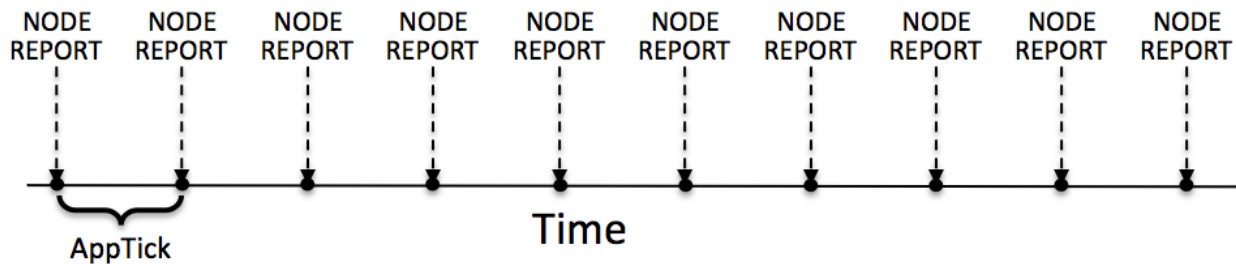
```
Terminal — uXMS — 146x24

VarName      (S) (T) VarValue (MODE = HISTORY:PAUSED)
-----
(1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=123.30,UTC_TIME=1281558263.13,X=0.00,Y=0.00,LAT=43.825300,LON=-70.330400,S
PD=0.00,HDG=180.00,YAW=180.00000,DEPTH=0.00,LENGTH=4.0,MODE=DISENGAGED,ALLSTOP=ManualOverride"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=124.30,UTC_TIME=1281558263.63,X=0.00,Y=0.00,LAT=43.825300,LON=-70.330400,S
PD=0.00,HDG=180.00,YAW=180.00000,DEPTH=0.00,LENGTH=4.0,MODE=DISENGAGED,ALLSTOP=ManualOverride"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=124.30,UTC_TIME=1281558264.13,X=0.00,Y=0.00,LAT=43.825300,LON=-70.330400,S
PD=0.00,HDG=180.00,YAW=180.00000,DEPTH=0.00,LENGTH=4.0,MODE=DISENGAGED,ALLSTOP=ManualOverride"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=125.30,UTC_TIME=1281558264.63,X=0.00,Y=0.00,LAT=43.825300,LON=-70.330400,S
PD=0.00,HDG=180.00,YAW=180.00000,DEPTH=0.00,LENGTH=4.0,MODE=DISENGAGED,ALLSTOP=ManualOverride"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=125.30,UTC_TIME=1281558265.13,X=0.00,Y=0.00,LAT=43.825300,LON=-70.330400,S
PD=0.00,HDG=180.00,YAW=180.00000,DEPTH=0.00,LENGTH=4.0,MODE=ENGAGED,ALLSTOP=clear"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=126.30,UTC_TIME=1281558265.63,X=0.00,Y=-0.60,LAT=43.825295,LON=-70.330400,
SPD=1.99,HDG=174.19,YAW=174.19226,DEPTH=0.23,LENGTH=4.0,MODE=MODE@ACTIVE:SURVEYING,ALLSTOP=clear"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=126.30,UTC_TIME=1281558266.13,X=0.13,Y=-1.59,LAT=43.825286,LON=-70.330398,
SPD=1.99,HDG=165.72,YAW=165.71888,DEPTH=0.60,LENGTH=4.0,MODE=MODE@ACTIVE:SURVEYING,ALLSTOP=clear"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=127.30,UTC_TIME=1281558266.63,X=0.40,Y=-2.55,LAT=43.825277,LON=-70.330394,
SPD=1.99,HDG=158.68,YAW=158.67596,DEPTH=0.98,LENGTH=4.0,MODE=MODE@ACTIVE:SURVEYING,ALLSTOP=clear"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=127.30,UTC_TIME=1281558267.13,X=0.79,Y=-3.47,LAT=43.825269,LON=-70.330390,
SPD=2.00,HDG=152.82,YAW=152.82229,DEPTH=1.35,LENGTH=4.0,MODE=MODE@ACTIVE:SURVEYING,ALLSTOP=clear"
NODE_REPORT_LOCAL (1) "NAME=henry,TYPE=UUV,MOOSDB_TIME=128.31,UTC_TIME=1281558267.63,X=1.26,Y=-4.35,LAT=43.825261,LON=-70.330383,
SPD=2.00,HDG=147.79,YAW=147.78912,DEPTH=1.73,LENGTH=4.0,MODE=MODE@ACTIVE:SURVEYING,ALLSTOP=clear"
```

The pNodeReporter Utility: The Optional Blackout Interval Option

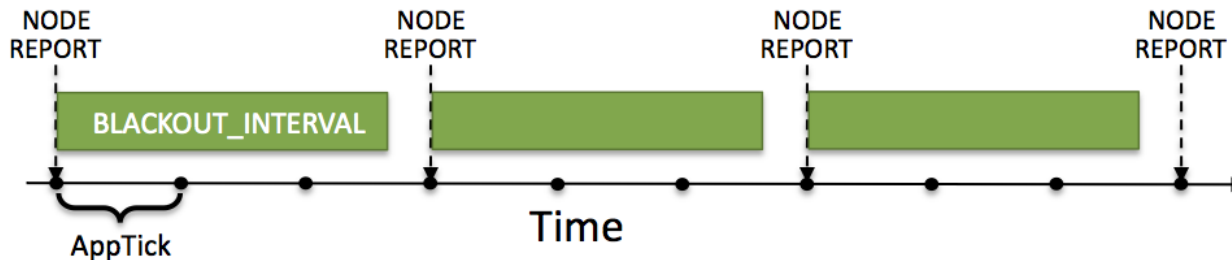


Normally a node report once per iteration, determined solely by the APP_TICK parameter.



At times it is useful to add an artificial delay between postings.

`BLACKOUT_INTERVAL = 35`



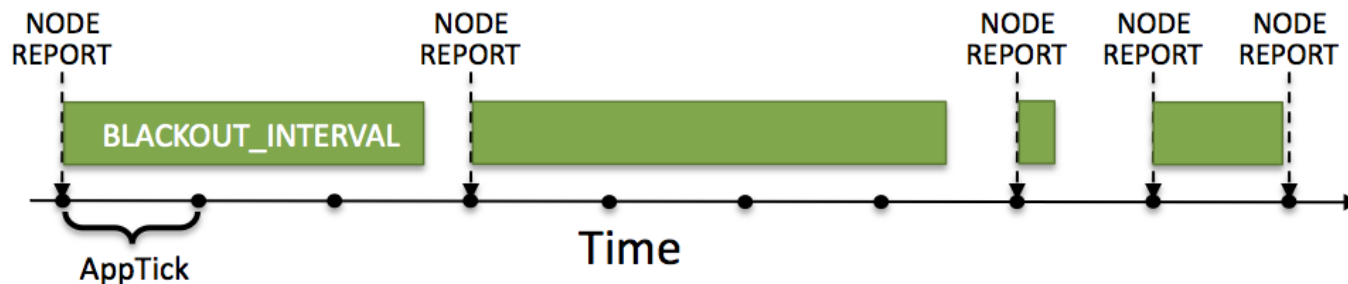
The pNodeReporter Utility: Random Blackout Intervals



- Node reports are typically only useful as information sent to other nodes.
- There are often dropped node messages due to the uncertain nature of communications.
- Applications receiving node reports usually implement provisions that take dropped messages into account.
- For example, a collision avoidance behavior may extrapolate the contact position in between node reports.
- To test the robustness of dealing with dropped node reports, we want to simulate them easily.
- The dropouts occur in the field more or less randomly (but may be range dependent etc.)

The Blackout Interval may be configured to vary randomly:

```
BLACKOUT_VARIANCE = 45
```





The uHelmScope Utility: Scoping on the IvP Helm



MOOS Modules:

- uXMS - A tool for focused scoping of the MOOSDB from the console.
- uPokeDB - A tool for poking the MOOSDB from the command line.
- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string.
- uHelmScope - A specialized scope on IvP Helm status and recent history.
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts.
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB.



The uHelmScope Utility: What it is, and is not



What is uHelmScope?

- It is a specialized scope on the MOOSDB for reporting information specific to the IvP Helm.
- It is console-based (like uXMS) and requires no graphics libraries.
- It reports on which behaviors are active, running, idle and complete.
- It reports the helm decision for each decision variable.
- It is capable of pausing and stepping back and forth in time.
- It includes a generic MOOS scope for convenience.
- It reports on which variables are posted by the helm on a given iteration.

What uHelmScope is NOT:

- It does not post any information to the helm or any other MOOS applications.
- It is not a graphical tool.

The uHelmScope Utility: A typical report from uHelmScope



From the Alpha example mission, shortly after deploying the vehicle:

```

Terminal - uHelmScope -- 127x38 -- 66
===== uHelmScope Report ===== ENGAGED (2)
Helm Iteration: 403 (hz=0.25)(5) (hz=0.25)(18) (hz=0.26)(max)
IvP functions: 1
Mode(s):
SolveTime: 0.00 (max=0.00)
CreateTime: 0.00 (max=0.01)
LoopTime: 0.00 (max=0.01)
Halted: false (0 warnings)
Helm Decision: [speed,0,4,21] [course,0,359,360]
course = 73.0
speed = 2.0
Behaviors Active: ----- (1)
waypt_survey (100.9) (pwt=100.00) (pcs=6) (cpu=0.43) (upd=0/0)
Behaviors Running: ----- (0)
Behaviors Idle: ----- (2)
waypt_return, hsline
Behaviors Completed: ----- (0)

# MOOSDB-SCOPE ----- (Hit '#' to en/disable)
#
# VarName Source Time Community VarValue
# -----
# BHV_WARNING n/a n/a n/a n/a
# DEPLOY* pMarin..ewer 128.28 alpha "true"
# HSLINE* pHelmIvP 128.28 alpha "off"
# RETURN* pMarin..ewer 128.28 alpha "false"

@ BEHAVIOR-POSTS TO MOOSDB ----- (Hit '@' to en/disable)
@
@ MOOS Variable Value
@ ----- (BEHAVIOR=waypt_survey)
@ WPT_STAT vname=alpha,behavior-name=waypt_survey,index=2,hit +
@ VIEW_POINT active,true:label,alpha_waypoint:label_color,0.000 +
@ VIEW_POINT active,true:label,alpha_track-point:label_color,0. +

```

The *Helm Report*: An overview of which behaviors' run state and information on IvP function characteristics and solve time.

The *MOOS Scope*: A mini MOOS scope for the convenience of scoping on a few variables of the user's choosing.

The *Behavior-Posts*: A set of variable-value pairs posted by the helm in the current iteration.



The uHelmScope Utility: A Closer Look at the Helm Report (the top section)

of IvP functions
in current decision

Engagement
status

of reports
written to
the console

```

===== uHelmScope Report ===== ENGAGED (2)
Helm Iteration: 403      (hz=0.25)(5)  (hz=0.25)(18)  (hz=0.26)(max)
IvP functions: 1
Mode(s):
SolveTime:      0.00      (max=0.00)
CreateTime:     0.00      (max=0.01)
LoopTime:       0.00      (max=0.01)
Halted:         false    (0 warnings)
Helm Decision: [speed,0,4,21] [course,0,359,360]
               course = 73.0
               speed = 2.0
Behaviors Active: ----- (1)
   waypt_survey (100.9) (pwt=100.00) (pcs=6) (cpu=0.43) (upd=0/0)
Behaviors Running: ----- (0)
Behaviors Idle: ----- (2)
   waypt_return, hsline
Behaviors Completed: ----- (0)

```

Average CPU
time between
iterations for the
last 5 iterations

Maximum CPU
time observed
for all iterations.

The uHelmScope Utility: A Closer Look at the Helm Report (the top section)

of IvP functions
in current decision

Engagement
status

of reports
written to
the console

```

===== uHelmScope Report ===== ENGAGED (2)
Helm Iteration: 403      (hz=0.25)(5)  (hz=0.25)(18)  (hz=0.26)(max)
IvP functions: 1
Mode(s):
SolveTime:      0.00      (max=0.00)
CreateTime:     0.00      (max=0.01)
LoopTime:       0.00      (max=0.01)
Halted:         false    (0 warnings)
Helm Decision: [speed,0,4,21] [course,0,359,360]
● course = 73.0
● speed = 2.0
Behaviors Active: ----- (1)
  waypt_survey (100.9) (pwt=100.00)
Behaviors Running: ----- (0)
Behaviors Idle: ----- (2)
  waypt_return, hsline
Behaviors Completed: ----- (0)
  
```

Total *solve* time for the current iteration –
and max solve time for all iterations.

Total *create* time for the current iteration
– and max solve time for all iterations.

The Helm decision space: variable name, low
value, high value and number of points.

The current helm decision.

The uHelmScope Utility:

A Closer Look at the Helm Report (the top section)

```

===== uHelmScope Report ===== ENGAGED (2)
Helm Iteration: 403      (hz=0.25)(5)  (hz=0.25)(18)  (hz=0.26)(max)
IvP functions: 1
Mode(s):
SolveTime:      0.00      (max=0.00)
CreateTime:     0.00      (max=0.01)
LoopTime:       0.00      (max=0.01)
Halted:         false     (0 warnings)
Helm Decision: [speed,0,4,21] [course,0,359,360]
  course = 73.0
  speed = 2.0
Behaviors Active: ----- (1)
  waypt_survey (100.9) (pwt=100.00) (pcs=6) (cpu=0.43) (upd=0/0)
Behaviors Running: ----- (0)
Behaviors Idle: ----- (2)
  waypt_return, hsline
Behaviors Completed: ----- (0)
  
```

Total behaviors in this run state

CPU time to make the IvP Function

Priority Weight

Pieces in the IvP Function

of successful updates vs. # of attempted behavior updates.

Behavior States: Which behaviors are active, running, idle or completed. For active behaviors, information is given on their IvP function.



The uHelmScope Utility:

A Closer Look at the MOOS Scope (middle section)

```
# MOOSDB-SCOPE ----- (Hit '#' to en/disable)
#
# VarName          Source          Time          Community    VarValue
# -----
# BHV_WARNING      n/a              n/a           n/a          n/a
# DEPLOY*          pMarin..ewer    128.28       alpha        "true"
# HSLINE*          pHelmIvP        128.28       alpha        "off"
# RETURN*          pMarin..ewer    128.28       alpha        "false"
```

List of variables to scope

Source of the last post

Time of the last post

Variable Value



The uHelmScope Utility:

A Closer Look at the MOOS Scope (middle section)

The *Behavior-Posts* section displays only those variable-value pairs posted by the Helm on the current iteration.

```
@ BEHAVIOR-POSTS TO MOOSDB ----- (Hit '@' to en/disable)
@
@ MOOS Variable      Value
@ -----          ----- (BEHAVIOR=waypt_survey)
@ WPT_STAT           vname=alpha,behavior-name=waypt_survey,index=2,hit +
@ VIEW_POINT         active,true:label,alpha_waypoint:label_color,0.000 +
@ VIEW_POINT         active,true:label,alpha_track-point:label_color,0. +
@
```

Variables posted

Variable Values

The uHelmScope Utility: Examining the Helm Hierarchical Mode Declarations



The Hierarchical Mode Declarations for Henry in the Berta Example Mission:

```
// Excerpt from the
// henry.bhv file.

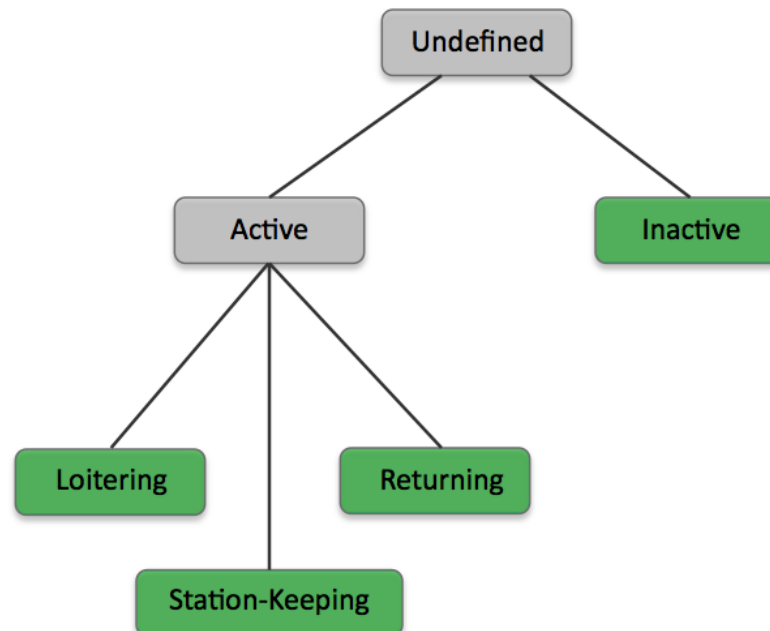
Set MODE = Active {
  DEPLOY = true
} Inactive

Set MODE = STATION-KEEPING {
  MODE = ACTIVE
  STATION_KEEP = true
}

Set MODE = RETURNING {
  MODE = ACTIVE
  RETURN = true
}

Set MODE = LOITERING {
  MODE = ACTIVE
  LOITER = true
}
```

Text File Configuration



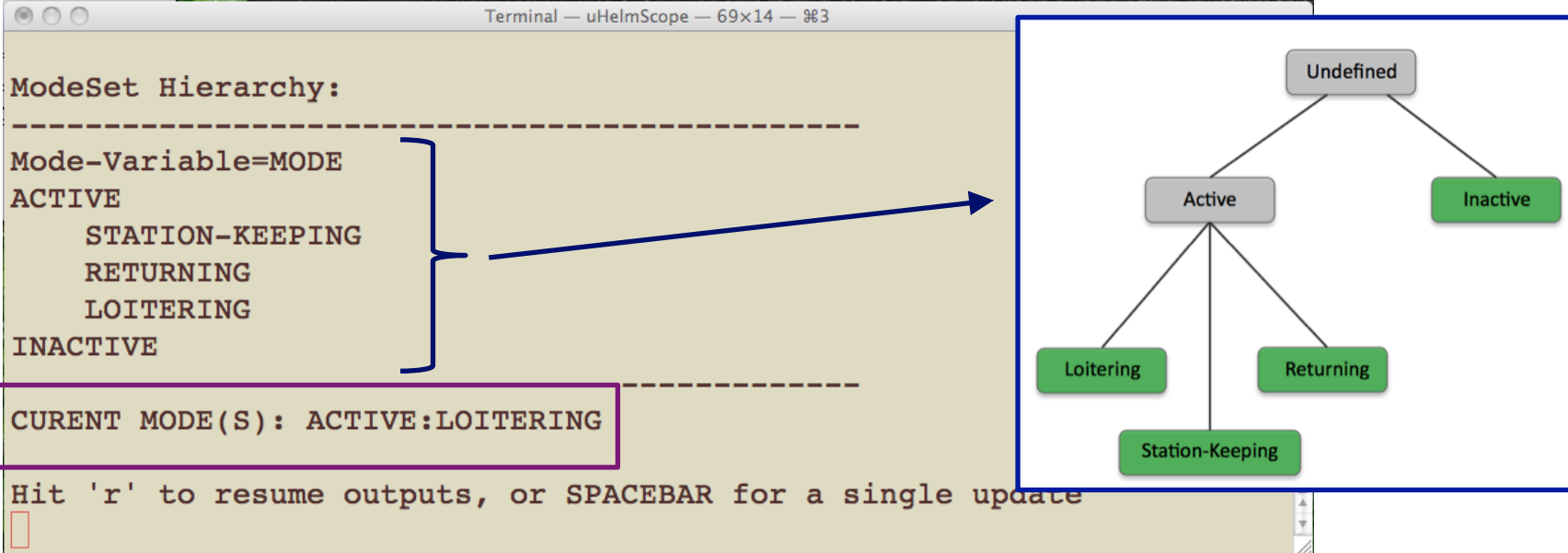
Graphical representation (manually generated)

uHelmScope can be used to visually confirm the configuration matches the intention.

The uHelmScope Utility: Examining the Helm Hierarchical Mode Declarations



The hierarchical mode declarations may be viewed by toggling with the 'M' key:



The terminal window displays the following text:

```

Terminal — uHelmScope — 69x14 — ⌘3
ModeSet Hierarchy:
-----
Mode-Variable=MODE
ACTIVE
  STATION-KEEPING
  RETURNING
  LOITERING
INACTIVE
-----
CURRENT MODE(S): ACTIVE:LOITERING
Hit 'r' to resume outputs, or SPACEBAR for a single update
  
```

A tree diagram to the right illustrates the hierarchy:

- Undefined
 - Active
 - Loitering
 - Station-Keeping
 - Returning
 - Inactive

Annotations in the image include a blue arrow pointing from the 'ACTIVE' section of the terminal to the 'Active' node in the tree, and a purple box around 'CURRENT MODE(S): ACTIVE:LOITERING' with an arrow pointing to the explanatory text below.

The prevailing helm mode is also shown in this screen.



The pBasicContactMgr Utility: Managing Platform Contacts



MOOS Modules:

- uXMS - A tool for focused scoping of the MOOSDB from the console.
- uPokeDB - A tool for poking the MOOSDB from the command line.
- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string.
- uHelmScope - A specialized scope on IvP Helm status and recent history.
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts.
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB.



The pBasicContactMgr Utility: What it is, and is not

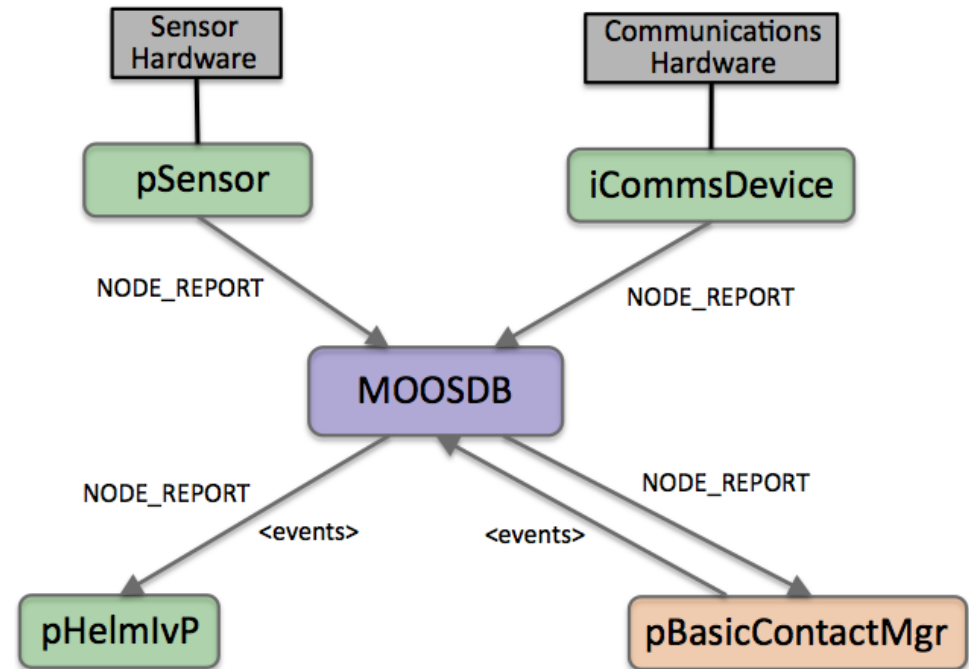


What is pBasicContactMgr?

- A tool for managing node reports and generating conditional events.
- It posts summary reports for all known contacts.
- It posts events, i.e., alerts, about contacts based on the range to the contact.
- Designed with the IvP Helm in mind to allow the helm to spawn contact-related behaviors dynamically as they become known.

What pBasicContactMgr is NOT:

- It is not a sensor application.
- It does not perform sensor fusion.
- It does not represent or reason about areas of uncertainty associated with contact position.



Variables Published:

- CONTACTS_LIST
- CONTACTS_RECAP
- CONTACT_ALERTED
- CONTACTS_UNALERTED
- CONTACTS_RETIRED
- CONTACT_MGR_WARNING



The pBasicContactMgr Utility: Alerts



What is an Alert?

- It is a posting to the MOOSDB – A MOOS variable-value pair.
- Alerts are generated for a given contact, when the contact is within a given range.
- The value of the alert is configured by the user in the pBasicContactMgr configuration block.

How are they used?

- Alerts may be used to trigger other processes.
- Alerts may also be used for marking an event to be logged and later referenced.

An example (collision avoidance):

- An alert is generated when contact gets “too close”.
- The helm is configured with a collision avoidance behavior “template”.
- The template is instantiated with a new behavior instance when it receives the alert.



The pBasicContactMgr Utility: Alert Configuration



Alerts are configured in the MOOS configuration file:

```
ALERT = var=<moos-variable>, val=<alert-content>
```

The <alert-content> may be any string, including certain macros for expansion. For example:

```
ALERT = var=CONTACT_INFO, val="name=avd_${VNAME} # contact=${VNAME}"
```

Macros available are:

(many of the same fields found in the node reports, NODE_REPORT)

- [\$VNAME]: The name of the contact.
- [\$X]: The position of the contact in local x coordinates.
- [\$Y]: The position of the contact in local y coordinates.
- [\$LAT]: The latitude position of the contact in earth coordinates.
- [\$LON]: The longitude position of the contact in earth coordinates.
- [\$HDG]: The reported heading of the contact.
- [\$SPD]: The reported speed of the contact.
- [\$DEP]: The reported depth of the contact.
- [\$VTYPE]: The reported vessel type of the contact.
- [\$UTIME]: The UTC time of the last report for the contact.



The pBasicContactMgr Utility: Alert Configuration



Alerts are configured in the MOOS configuration file:

```
ALERT = var=<moos-variable>, val=<alert-content>
```

The <alert-content> may be any string, including certain macros for expansion. For example:

```
ALERT = var=CONTACT_INFO, val="name=avd_${VNAME} # contact=${VNAME}"
```

Macros available are:

Accommodates a helm behavior configuration for dynamic behavior spawning. See the Berta example mission.

(many of the same fields found in the node reports, NODE_REPORT)

- \${VNAME}: The name of the contact.
- \${X}: The position X coordinate.
- \${Y}: The position Y coordinate.
- \${LAT}: The latitude.
- \${LON}: The longitude.
- \${HDG}: The reported heading of the contact.
- \${SPD}: The reported speed of the contact.
- \${DEP}: The reported depth of the contact.
- \${VTYPE}: The reported vessel type of the contact.
- \${UTIME}: The UTC time of the last report for the contact.

The pBasicContactMgr Utility: Alert Triggers



Alerts are triggered by range. Configured in the MOOS configuration file:

```

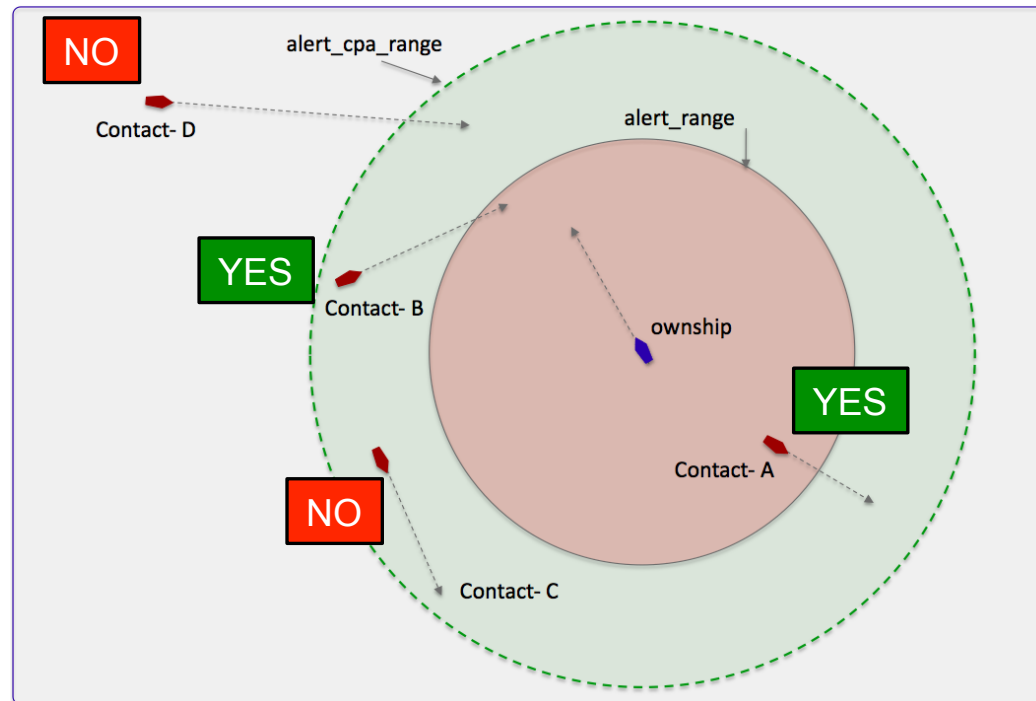
ALERT_RANGE = <distance> // meters
ALERT_CPA_RANGE = <distance> // meters
ALERT_CPA_TIME = <duration> // seconds
    
```

ALERT_RANGE – when a contact is within this range an alert is generated.

ALERT_CPA_RANGE – when a contact is within this range and its closest point of approach (CPA) is within the alert range, an alert is generated.

ALERT_CPA_TIME – The time used for CPA calculation.

Examples:





The pBasicContactMgr Utility: Contacts, Alerts, Record keeping



The following are reported (Posted to the MOOSDB) on each iteration:

- CONTACTS_LIST: comma-separated list of contacts.
- CONTACTS_RECAP: A comma-separated list of contact summaries.
- CONTACT_ALERTED: A list of contacts for which alerts have been posted.
- CONTACTS_UNALERTED: A list of contacts for which alerts are pending, based on the range criteria.
- CONTACTS_RETIRED: A list of contacts removed due to the information staleness.
- CONTACT_MGR_WARNING: A warning message indicating possible mishandling of or missing data.

Examples:

- CONTACTS_LIST: = "delta,gus,charlie,henry"
- CONTACT_ALERTED: = "delta,charlie"
- CONTACTS_UNALERTED: = "gus,henry"
- CONTACTS_RETIRED: = "bravo,foxtrot,kilroy"
- CONTACTS_RECAP: = "name=delta,age=11.3,range=193.1 # name=gus,age=0.7,range=48.2
#name=charlie,age=1.9,range=73.1 # name=henry,age=4.0,range=18.2"



The pBasicContactMgr Utility: Contact Resolution



- An alert is generated by the contact manager for a given contact ONCE (when the trigger criteria is first met).
- Sometimes a consumer of alerts may want to receive additional future alerts should the contact come back into range.
- If pBasicContactMgr receives the message `CONTACT_RESOLVED`, for a given contact, it will generate another alert for that contact should the contact again meet the trigger criteria.

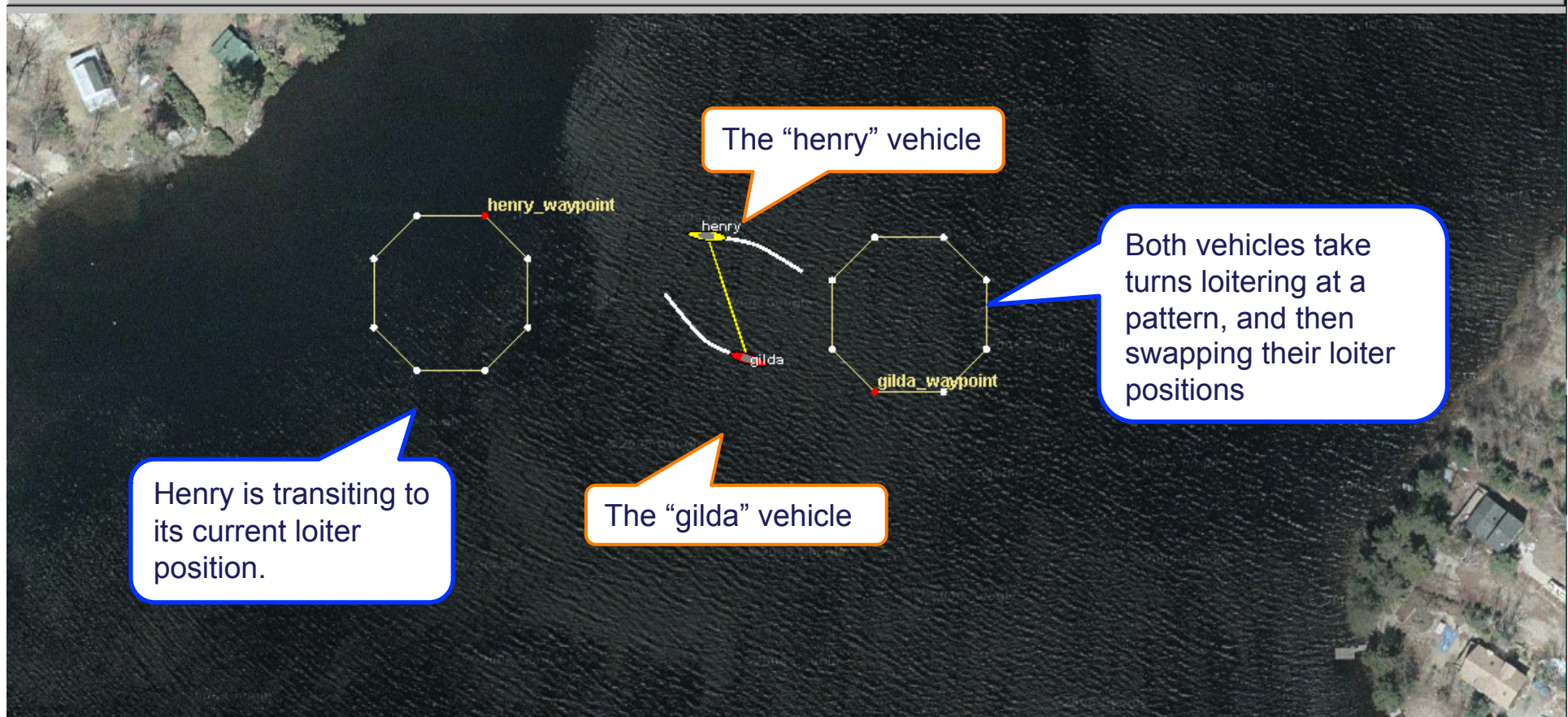
The contact resolution mechanism is used to handle the scenario where a contact comes into range, exits the range, and later returns.

The pBasicContactMgr Utility: The Berta Example Mission



pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope ReferencePoint Action



The “henry” vehicle

The “gilda” vehicle

Henry is transiting to its current loiter position.

Both vehicles take turns loitering at a pattern, and then swapping their loiter positions

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

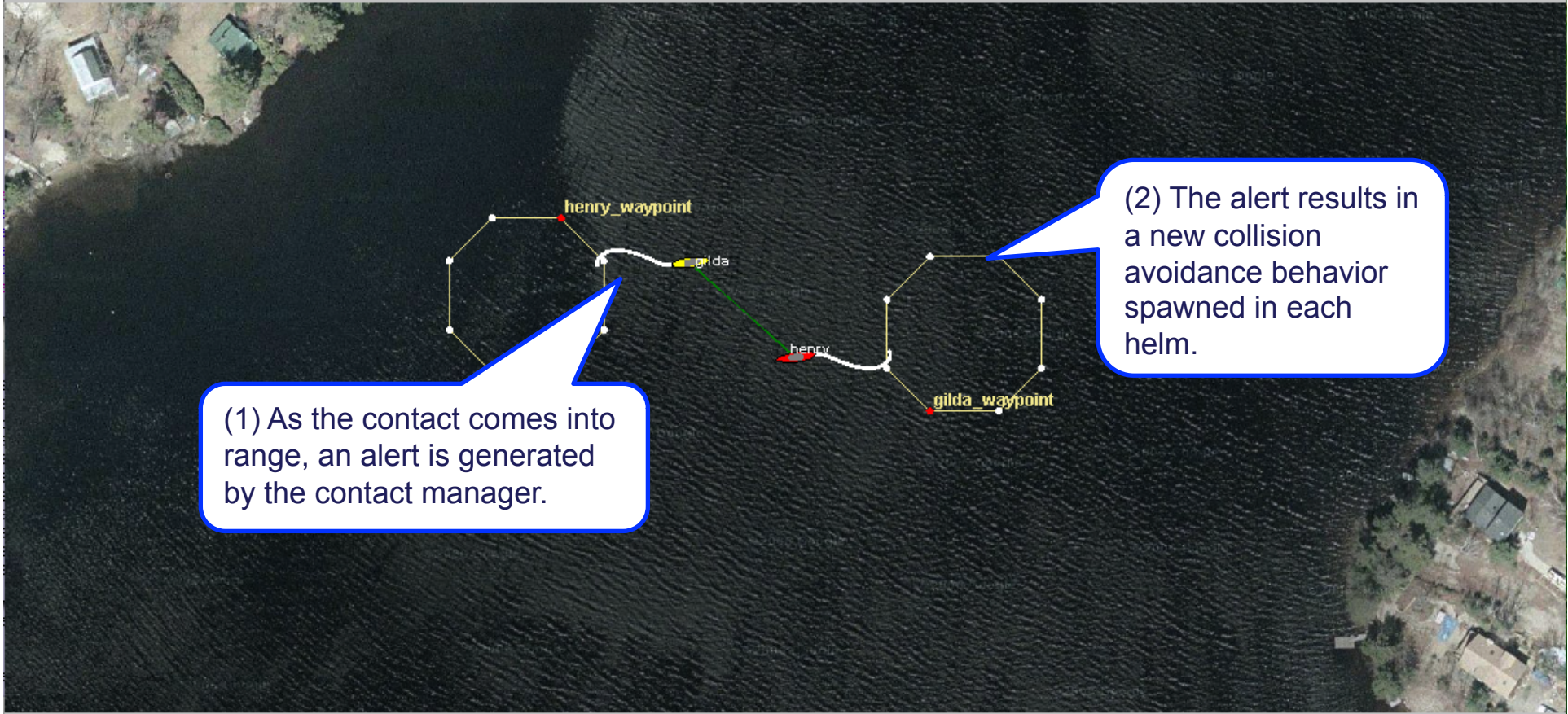
Variable: Time: Value:

The pBasicContactMgr Utility: The Berta Example Mission



pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope ReferencePoint Action



(1) As the contact comes into range, an alert is generated by the contact manager.

(2) The alert results in a new collision avoidance behavior spawned in each helm.

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

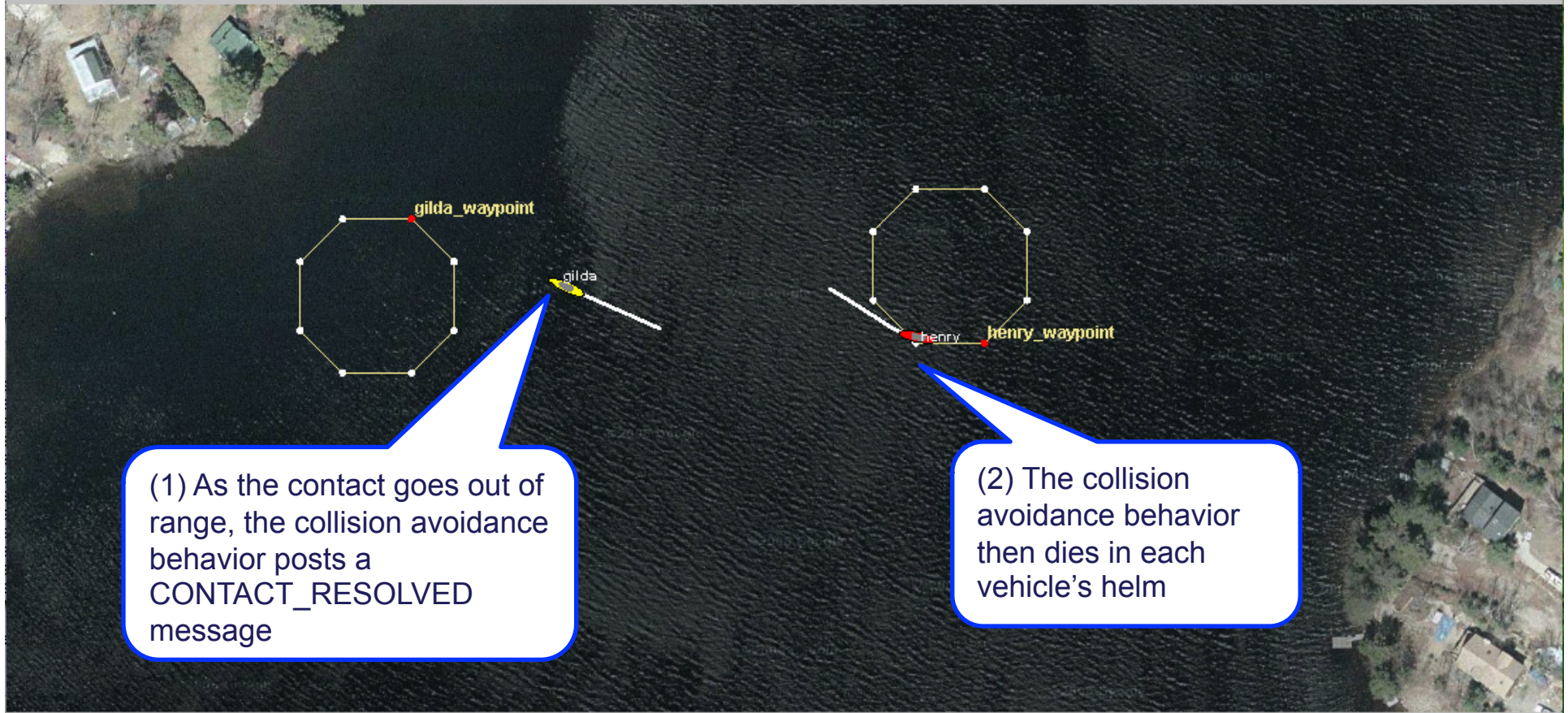
Variable: Time: Value:

The pBasicContactMgr Utility: The Berta Example Mission



pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope ReferencePoint Action



(1) As the contact goes out of range, the collision avoidance behavior posts a CONTACT_RESOLVED message

(2) The collision avoidance behavior then dies in each vehicle's helm

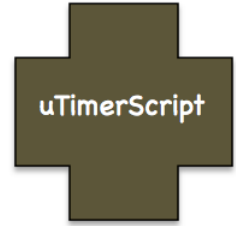
VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:



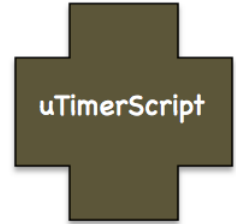
The uTimerScript Utility: Scripting Events to the MOOSDB



MOOS Modules:

- uXMS - A tool for focused scoping of the MOOSDB from the console.
- uPokeDB - A tool for poking the MOOSDB from the command line.
- pMarineViewer - A GUI tool for rendering vehicle operations onto an geo-referenced display.
- pNodeReporter - Captures vehicle state information and publishes a summary string.
- uHelmScope - A specialized scope on IvP Helm status and recent history.
- pBasicContactMgr - A simple manager of vehicle contacts, and generation of alerts.
- uTimerScript - A tool for scripting (possibly conditional and random) pokes to the MOOSDB.

The uTimerScript Utility: Overview



What is uTimerScript?

- A tool that allows the user to script a set of pre-configured events (pokes) to a MOOSDB.
- Each event can be configured to happen after a specified amount of elapsed time.
- Enables us to fake incoming command-and-control messages, sensor events etc.

A simple example:

```
ProcessConfig = uTimerScript
{
  AppTick      = 4
  CommsTick    = 4

  EVENT = MOOS_MANUAL_OVERRIDE, false, 10
  EVENT = var=DEPLOY, val=true, time=15
}
```

This simple script will launch the Alpha or Berta missions automatically.

EVENT = var<variable>, val=<value>, time=<delay>



The uTimerScript Utility: Starting and Pausing the Script



When does the script start?

- By default the script starts when uTimerScript connects to the MOOSDB and begins to Iterate().
- It may be configured in the “paused” mode
- It may be configured to include a delay once it has started.
- It may be configured to require conditions be met before starting.

Starting the script in the PAUSED mode, with a DELAY.

```
ProcessConfig = uTimerScript
{
  AppTick      = 4
  CommsTick    = 4

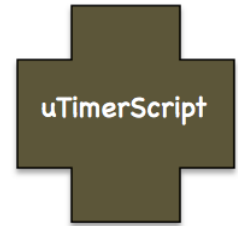
  EVENT = var=MOOS_MANUAL_OVERRIDE, val=false, time=10
  EVENT = var=DEPLOY, val=true, time=15

  CONDITION = ALPHA != 20
  DELAY_START = 30
  PAUSED = true
}
```

Script will be paused if ALPHA=20.
(uTimerScript will register for ALPHA) .

The script may then be un-paused by posting to the MOOSDB:
UTS_PAUSE=false.

The uTimerScript Utility: Randomizing the Event Times



Random event scheduling:

- Events may be configured to occur at a random time in a given interval.
- Random events are useful in testing the robustness of algorithms in varying situations.

The same example script with events randomized:

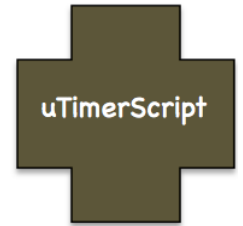
```
ProcessConfig = uTimerScript
{
  AppTick      = 4
  CommsTick    = 4

  EVENT = var=MOOS_MANUAL_OVERRIDE, val=false, time=10:20
  EVENT = var=DEPLOY, val=true, time=10:20
  PAUSED = true
}
```

Event occurs between
10 and 20 seconds after
the script begins

Event times are chosen with uniform probability.

The uTimerScript Utility: Repeating the script



Repeating the script:

- The script may be repeated a fixed number or indefinite number of times.

The same example script with events randomized:

```
ProcessConfig = uTimerScript
{
  AppTick      = 4
  CommSTick    = 4

  EVENT = var=MOOS_MANUAL_OVERRIDE, val=false, time=10:20
  EVENT = var=DEPLOY, val=true, time=10:20

  RESET_MAX = 10
  RESET_TIME = 120
  RESET_VAR = UTS_RESET
  DELAY_RESET = 12
  SHUFFLE = true
}
```

Script will run 11 times.

Script will reset after 120 seconds regardless if it is finished.

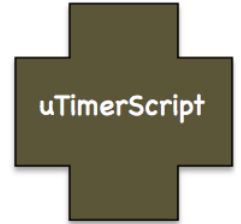
MOOS variable for receiving reset cues.

of seconds the script will delay on each reset.

If shuffle is false, random timestamps will not be recalculated on each reset.



The uTimerScript Utility: Macro Expansion



Macros:

- Macros are used to fill in variable values with information determined at event posting time.

A Script with a simple macro posting:

```
ProcessConfig = uTimerScript
{
  AppTick      = 4
  CommstTick   = 4

  EVENT = var=MOOS_MANUAL_OVERRIDE, val=false, time=10:20
  EVENT = var=DEPLOY, val=true, time=10:20
  EVENT = var=SCRIPT_STARTED, val=${DBTIME}, time=0
}
```

The start time of the script will be posted with the value of DBTIME, the total amount of time the MOOSDB has been up.

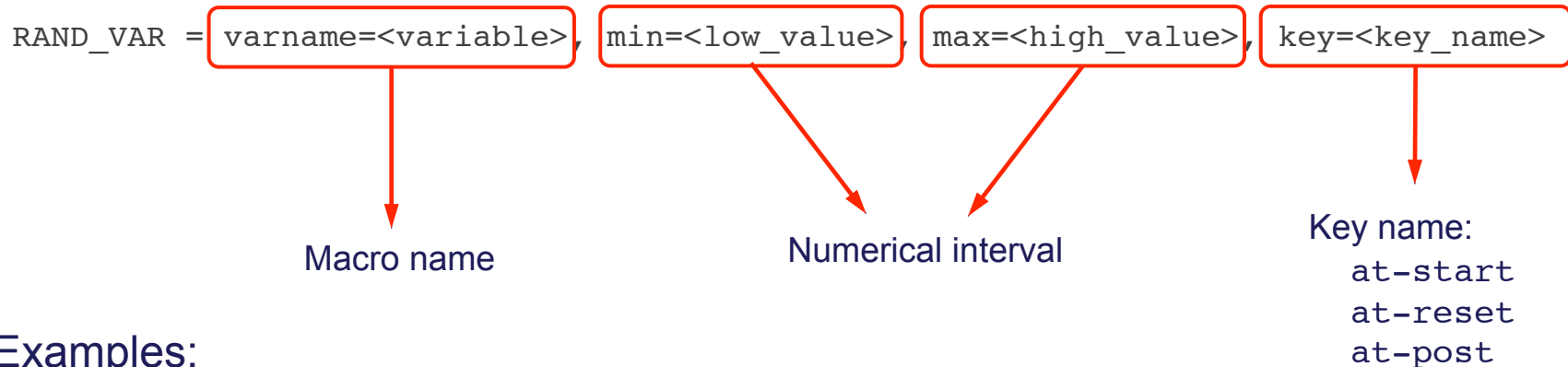
The uTimerScript Utility: Macro Expansion



Macros available:

- `${DBTIME}`: The estimated amount of time since the MOOSDB started.
- `${UTCTIME}`: The UTC time at the time of event posting.
- `${COUNT}`: The integer total of all posts thus far in the script – reset to zero on script reset.
- `${TCOUNT}`: Same as above except the total is not reset when the script is reset.
- `${IDX}`: Similar to `${COUNT}`, but it expands as a string, “000”, “001”, “002”, etc.

User configured macros with random variables:



Examples:

```
RAND_VAR = varname=ANGLE, min=0, max=359, key=at_reset
```

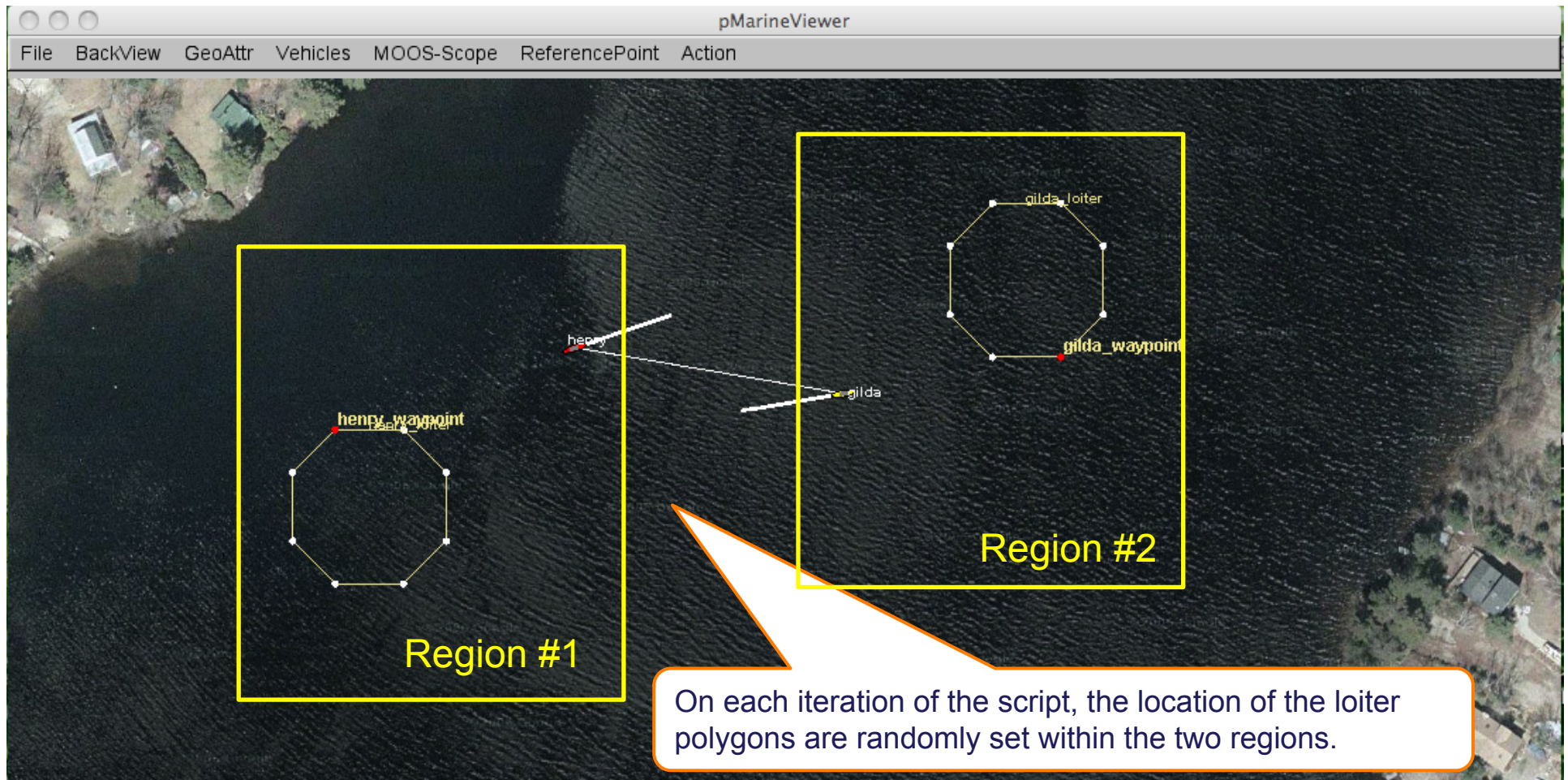
```
RAND_VAR = varname=MAGNITUDE, min=0.5, max=1.5, key=at_reset
```

The uTimerScript Utility: Usage in the Berta Example Mission



pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope ReferencePoint Action



Region #1

Region #2

On each iteration of the script, the location of the loiter polygons are randomly set within the two regions.

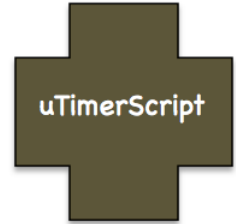
VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:



The uTimerScript Utility: Usage in the Berta Example Mission



pMarineViewer

File BackView GeoAttr Vehicles MOOS-Scope ReferencePoint Action

Region #1

Region #2

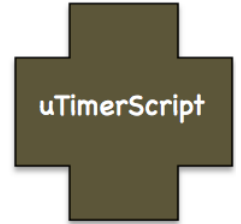
On each iteration of the script, the location of the loiter polygons are randomly set within the two regions.

VName: X(m): Lat: Spd(m/s): Dep(m): Time: Range:

VType: Y(m): Long: Heading: Report-Age: Warp: Bearing:

Variable: Time: Value:

The uTimerScript Utility: Script Usage in the Berta Example Mission



Permutation of Region locations and loiter assignments

```
ProcessConfig = uTimerScript
```

```
{
```

```
  AppTick    = 4
```

```
  CommsTick  = 4
```

```
  PAUSED     = false
```

```
  RESET_MAX  = unlimited
```

```
  RESET_TIME = end
```

```
  RANDVAR = varname=X1, min=-25, max=25, key=at_reset
```

```
  RANDVAR = varname=Y1, min=-100, max=-50, key=at_reset
```

```
  RANDVAR = varname=X2, min=100, max=150, key=at_reset
```

```
  RANDVAR = varname=Y2, min=-75, max=-25, key=at_reset
```

```
  EVENT = var=UP_LOITER_2, val="center_assign=${X1},${Y1}", time=180
```

```
  EVENT = var=UP_LOITER_1, val="center_assign=${X2},${Y2}", time=180
```

```
  EVENT = var=UP_LOITER_1, val="center_assign=${X1},${Y1}", time=360
```

```
  EVENT = var=UP_LOITER_2, val="center_assign=${X2},${Y2}", time=360
```

```
}
```

Random variable Macro
for Region #1

Region #2

Macro usage in
scripted events



The End

Where to find more:

On the web:

www.moos-ivp.org

Email:

issues@moos-ivp.org