The uFldTagManager Application

Fall 2017

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

1 Overview

The `uFldTagManager` application is used for field management of inter-vehicle tags during head-to-head competitions. It is a key component of the *Aquaticus* competition, but could be used in other competitions as well. The application runs on the shoreside, accepts a tag request from a vehicle, applies the tag criteria, and either grants or denies the tag.

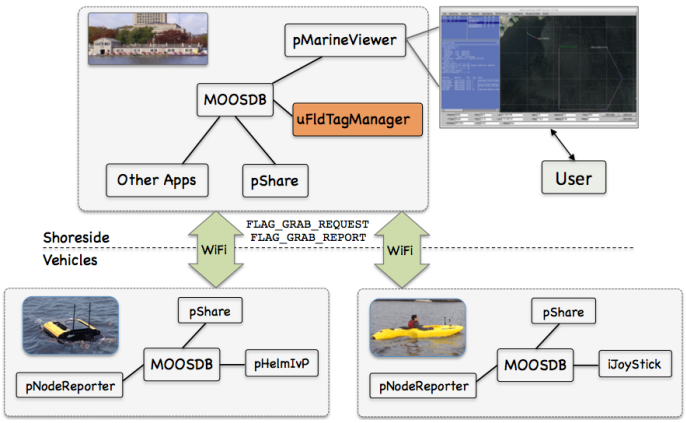

Figure 1: **Typical uFldTagManager Topology:** The tag manager runs on the shoreside and handles requests from one vehicle to tag another vehicle.

2 Configuration Parameters for uFldTagManager

The following parameters are defined for `uFldTagManager`. A more detailed description is provided in other parts of this section. Parameters having default values are indicated.

Listing 2.1: Configuration Parameters for `uFldTagManager`.

`human_platform`: The vehicle type, as seen on incoming node reports, to be regarded as human-controlled platform. The default is `"mokai"`. Section 4.4.

`human_tag_post`: A MOOS variable-value pair to be posted whenever at tag is applied to a human-controlled platform. Multiple pairs allowed. Section 4.4.

`human_untag_post`: A MOOS variable-value pair to be posted whenever at tag expires on a human-controlled platform . Multiple pairs allowed. Section 4.4.

`post_color`: The color of the tag visual upon initial posting, before a decision is made. Default is white. Section 4.2.

`robot_tag_post`: A MOOS variable-value pair to be posted whenever at tag is applied to a non human-controlled platform. Multiple pairs allowed. Section 4.4.

`robot_untag_post`: A MOOS variable-value pair to be posted whenever at tag expires on a non human-controlled (robot) platform. Multiple pairs allowed. Section 4.4.

`tag_circle`: If true, a small circle is posted by `uFldTagManager` to be rendered around the tagged vehicle. The default is true. Section 4.2.

`tag_circle_color`: Specifies the color of the tag circle. The default is green. Section 4.2.

`oob_circle_color`: Specifies the color of the tag circle, when the vehicle has been deemed tagged due to being out-of-bounds. The default is yellow. Section 4.2.

`tag_circle_range`: Specifies the range, in meters, of the circle to be rendered around a tagged vehicle. The default is 5 meters. Section 4.2.

`tag_duration`: The amount of time, in seconds, before a successfully applied tag expires. The default is 30 seconds. Section4.1.

tag_min_interval:	The time required after one tag, before another tag is allowed. The default is 10 seconds. Section 4.1.
tag_range:	The range (meters) within which a tag request is granted. Section 4.1.
team_one:	The name of team one. Section 4.3.
team_two:	The name of team two. Section 4.4.
zone_one:	A convex polygon representing the home region for team one. Section 4.3.
zone_one_color:	The color used for rendering zone one. The default is white. Use "empty" to specify no fill color. Section 4.3.
zone_two:	A convex polygon representing the home region for team two. Section 4.3.
zone_two_color:	The color used for rendering zone two. The default is green. Use "empty" to specify no fill color. Section 4.3.

3 Publications and Subscriptions of uFldTagManager

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

```
$ uFldTagManager --interface or -i
```

3.1 Variables Published by uFldTagManager

The following variables are published by uFldTagManager:

- 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. Section 6.
- TAG_RESULT_VNAME: A brief result description of the most recent tag request. Section 5.9.
- TAG_RESULT_VERBOSE: A more detailed result of the most recent tag request. Section 5.4.
- TAG_RELEASE_VERBOSE: The details of a release event, the expiration of a previously applied tag. Section 5.8.
- VIEW_RANGE_PULSE: A visual marker posted upon receipt of a particular tag request. Section 4.2.
- VIEW_CIRCLE: A visual circle posted to encircle a vehicle that is in a tagged state. By default green when tagged by an enemy, and yellow when tagged due to being out of bounds. Section 4.2.
- VIEW_POLYGON: A visual polygon, typically to represent the home zones of each team. Section 4.3.

Variables configured in the human_tag_post, robot_tag_post, human_untag_post, and robot_untag_post parameters will also be published. These MOOS variables are mission specific and configured by the user.

3.2 Variables Subscribed for by uFldTagManager

The uFldTagManager application will subscribe for the following four MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration. Section 6.
- TAG_REQUEST: A request from a particular vehicle, to apply a tag immediately. Section 5.4.
- NODE_REPORT: A vehicle position report expected regularly from each vehicle in the field. Section 5.1.

4 Configuration of uFldTagManager

4.1 Configuring the Tagging Range, Duration and Min Interval

The tagging *range* is the distance, in meters, within which tag requests are granted. The default is 25 meters. The tag *duration* is the number of seconds that a tagged vehicle remains tagged before returning to an untagged state. The default is 30 seconds. The tag *min interval* is the amount of time required before a vehicle can apply a tag after successfully applying a previous tag. The default value is 10 seconds.

```
tag_range        = 25    // The default is 25 meters.
tag_duration     = 30    // The default is 30 seconds.
tag_min_interval = 10    // The default is 10 seconds.
```

4.2 Configuring the Tagging Visuals

When a tag request is received from one of the vehicles, an initial visual marker is posted to indicate the tag is under consideration. The post is to the variable VIEW_RANGE_PULSE. This marker is in form of an expanding circle. The color of this circle can be set with the post_color parameter. The default is white.

```
post_color = white    // The default color is white
```

If a tag is indeed applied to a target vehicle, this may be indicated with the posting of a small circle centered on the vehicle. The posting is to the variable VIEW_CIRCLE. This feature is enabled with the tag_circle parameter. The default is true. The color of this circle can be configured with the tag_circle_color parameter. The default is green. The color of the circle may be different if the vehicle is tagged due to being out of bounds. The default for this color is yellow, but may be configured with the oob_circle_color paramaeter. The range of this circle can be configured with the tag_circle_range parameter. The default is 10 meters. For example:

```
tag_circle = true              // The default is true
tag_circle_color = white       // The default is green
oob_circle_color = red         // The default is yellow
tag_circle_range = 5           // The default is 10 meters.
```

4.3 Configuring the Vehicle Team Names and Zones

The uFldTagManager assumes the existence of two teams. These two teams must be declared and be different. This is done with the team_one and team_two configuration parameters. For example:

```
 team_one = red
 team_one = blue
```

Each vehicle known to uFldTagManager must be on one of these teams, or else a run warning will be posted. Assigning a vehicle to a team is done by configuring the group parameter in pNodeReporter for each vehicle. For example:

```
ProcessConfig = pNodeReporter
{
  ...
  group = blue
  ...
}
```

The uFldTagManager also assumes the existence of two team *zones*. The zones are convex polygons where the vertices are provided in the zone_one and zone_two configuration parameters. For example:

```
  zone_one = pts={0,-20:120,-20:120,-100:0,-100}
  zone_two = pts={0,-100:120,-100:120,-180:0,-180}
```

The above two lines result in two posts to the variable VIEW_POLYLGON, which when handled by pMarineViewer, results in the rendering of the two zones shown below in Figure 2.

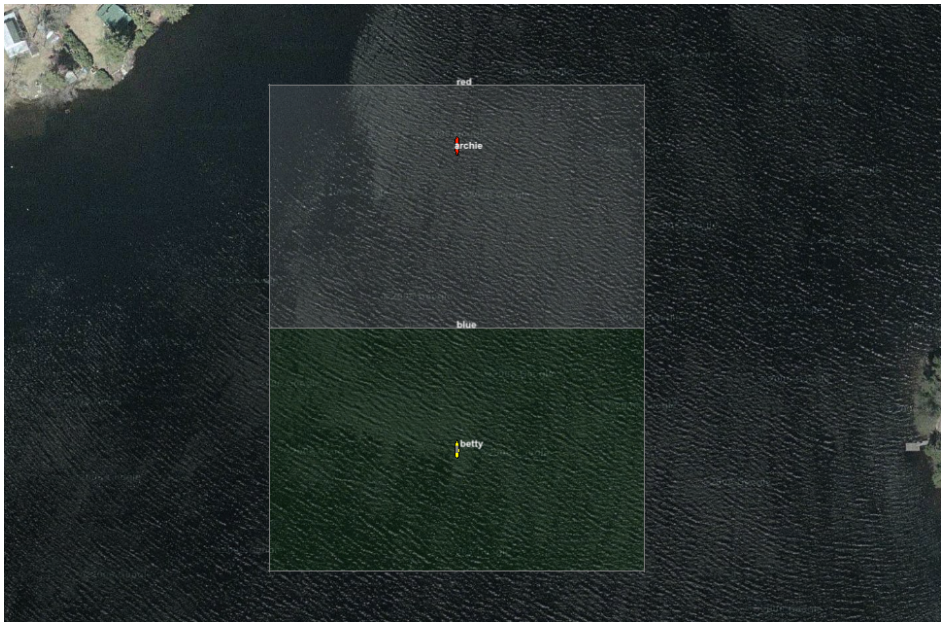

Figure 2: **Tag Manager Zones**: Example team zones from the alpha mission.

The user can also configure the internal colors of the two zones, or shut off the interal color completely, with the zone_one_color and zone_two_color parameters. For example:

```
zone_one_color = red     // The default is white
zone_two_color = blue    // The default is green
```

The internal color can be shut off completely with the special color name of "empty".

4.4 Configuring Tag and UnTag Postings

The tag manager may be configured to make user-specified postings to the MOOSDB upon a vehicle tag and upon a vehicle tag expiration. This is done with the robot_tag_post and robot_untag_post parameters respectively. For either type of post, single, multiple or no postings at all are perfectly legal configurations. Below is taken from the example mission:

```
robot_tag_post   = MOOS_MANUAL_OVERRIDE_$UP_TARGET=true
robot_tag_post   = SAY_MOOS=file=sounds/tennis_grunt.wav

robot_untag_post = MOOS_MANUAL_OVERRIDE_$UP_TARGET=false
robot_untag_post = SAY_MOOS=file=sounds/shipbell.wav
```

In the above case, every time a vehicle is tagged a message is posted to halt the tagged vehicle, and generate an audio cue. Whe the tag expires, the vehicle is allowed to resume its mission, and a different audio cue is posted.

If the tag is being applied to a human-controlled vehicle, the tag manager can be configure to post a different set of postings. This is done by swapping the "robot" for "human" in the configuration parameters. For example:

```
human_tag_post   = MESSAGE_TO_DRIVER="tagged"
human_untag_post = MESSAGE_TO_DRIVER="proceed"
```

To accomplish "vehicle specific" posts, certain macros are supported in specifying both the MOOS variable and the string value being posted. These macros are:

- $TARGET - the name of the vehicle just tagged, or whose tag has just expired.
- $UP_TARGET - the name of the vehicle just tagged, or whose tag has just expired. The vehicle name is converted to all upper case.
- $SOURCE - the name of the vehicle who just applied a tag.
- $UP_SOURCE - the name of the vehicle who just applied a tag. The vehicle name is converted to all upper case.
- $TIME - a UTC time stamp of the event.

The $SOURCE and $UP_SOURCE macros are only elevant only in the *_tag_post cases since there is no concept of a source when a tag expires. There is only one relevant vehicle in a tag expiration event, the vehicle originally targeted for a tag.

4.5 Determining Which Vehicles are Humans and Which are Robots

The tag managers gets vehicle information solely from node reports, typically generated by pNodeReporter. For example, the below node report indicates the report originated from a MOKAI (human-controlled) platform:

```
NODE_REPORT = "NAME=archie,X=0,Y=0,SPD=0,HDG=180,DEP=0,LAT=43.8253,LON=-70.3304,
               TYPE=mokai,GROUP=red,MODE=PARK,ALLSTOP=clear,INDEX=405,
               YAW=-1.5707963,TIME=1450007892.9,LENGTH=4
```

The vehicle type information is declared in the pNodeReporter configuration block. For example:

```
ProcessConfig pNodeReporter
{
  ...
  type = mokai
  ...
}
```

From the tag manager's perspective, it does not matter whether a source vehicle (the one applying the tag) is human or not. But it typically *does* matter whether the target vehicle (the one being tagged) is human or robot. This is because we may want to send a voice message to a human and a different kind of message to a robot to cue an appropriate behavior. The differences are present both in simulation and in the field. This is why there are separate configuration parameters for making tag posts, e.g., human_tag_post and robot_tag_post.

By default the tag manager regards vehicles of type "mokai" to be human-controlled platforms. This can be overridden by setting the human_platform configuration parameter to something else, e.g., human_platform=launch_boat.

5 Operation of uFldTagManager

5.1 Handling Node Reports

The tag manager needs to know where all the vehicles are, for both teams, in order to reason about and apply tags. This information comes via NODE_REPORT messages, originating on each vehicle, and shared via pShare to the shoreside where uFldTagManager is running. See Figure 1.

5.2 Ensuring Node Reports are Sent from Vehicles to Shore

The pNodeReporter application publishes its node report typically a few times per second in the MOOS variable NODE_REPORT_LOCAL. When this is shared to the shoreside, it changes its variable name to simply NODE_REPORT. Chances are, in any mission folder taken as a starting point or example, vehicles are already configured to share node reports to the shoreside (otherwise no vehicles would be visible in pMarineViewer. This sharing is configured in the vehicle configuration file, in uFldNodeBroker configuration block, with the line:

```
ProcessConfig uFldNodeBroker
{
  ...
  bridge = src=NODE_REPORT_LOCAL,  alias=NODE_REPORT
  ...
}
```

5.3 Configuring Team Membership for Vehicles

The tag manager also needs to know which *team* each vehicle belongs to. A vehicle's team is declared in the vehicle configuration file, in the pNodeReporter application, with a configuration line something like:

```
ProcessConfig pNodeReporter
{
  ...
  group = red
  ...
}
```

This group name, must match one of the team names declared in either the team_one or team_two configuration parameter of uFldTagManager (Section 4.3). If a group/team name is not included in the node report, uFldTagManager will post the run warning:

```
Node report for archie with no group
```

If a node report is received for a vehicle, where the vehicle group (team) is unknown to uFldTagManager, the following run warning will be posted:

```
Node report for betty w/ unknown team: purple
```

5.4 Handling Tag Requests

A tag request may originate from any vehicle known to the tag manager. The tag manager knows about a vehicle (its name, position and team membership) through incoming node reports. Based on a few criteria, described below, the tag manager may or may not grant the tag request. The *source* vehicle, requesting the tag, does not name a target vehicle. Thus the tag request is quite simple, of the form:

```
  TAG_REQUEST = vname=archie
```

The request simply identifies the requesting vehicle. A vehicle is prevented from making a request on behalf of another vehicle, by checking the MOOS community name associated with the incoming tag request. The community name and vname argument must match.

5.5 Criteria for Granting a Tag Request

For a *source* vehicle requesting a tag, to have its tag applied to a *target* vehicle, the following criteria must be met:

- *team membership*: The source vehicle and target vehicle must be on different teams.
- *zone*: The source vehicle must be within its own zone. The target vehicle must be outside his own zone (note - check that the latter is implemented this way).
- *frequency*: The source vehicle must wait at a minimum number of seconds, configured in the `tag_min_interval` configuration parameter, after a previous successful tag before a subsequent tag may be granted. Unsuccessful tag attempts have no effect on the clock.
- *target state*: The target vehicle must not be already tagged.
- *range*: The source to target range must be within the maximum range specified in the `tag_range` configuration parameter.
- *proximity*: If multiple qualifying targets are within range of the source vehicle, the target closest to the source is the one tagged.

5.6 The Results of a Tag Request (Posted to the MOOSDB)

When a tag request meets all the criteria described above, a tag is applied. The most significant consequences are user-define, in the form of MOOS postings configured in the configuration parameter `tag_post`. This is described further in Section 5.9.

Each time a tag is requested, regardless of the result or reason, a result is posted in the MOOS variable `TAG_RESULT`. It contains an event number, the source vehicle, the source vehicle's team, and the actual result. Below are a few examples:

```
TAG_RESULT_HENRY = "event=23,src=henry,team=red,rejected=freq"
TAG_RESULT_HENRY = "event=23,src=henry,team=red,rejected=zone"
TAG_RESULT_HENRY = "event=23,src=henry,team=red,tagged=none"
TAG_RESULT_HERFY = "event=23,src=henry,team=red,tagged=gilda"
```

Note the four possible results. If a vehicle tag *is* successful, as in the last posting, the tagged vehicle is named (`tagged=gilda`). A tag is *rejected* if the tag manager doesn't even consider the positions or state of nearby vehicles. The rejection may be due to either the vehicle being out of its home zone (`rejected=zone`), or because it has applied a previous tag too recently (`rejected=freq`). If the tag request is not rejected but there simply are no taggable vehicles within range, then the reported result is just `tagged=none`.

Note that the `TAG_RESULT` variable name is appended with the name of the source vehicle. The result information is to be sent to the source vehicle only. In a typical mission configuration, variables on the shoreside fitting the pattern of `FOOBAR_VNAME`, are configured to be shared out to the vehicle with name `VNAME`. The variable posted in vehicle's MOOSDB will have the name truncated, to `FOOBAR` in this case, with vehicle name suffix truncated.

The tag manager also publishes, for each received `TAG_REQUEST`, a verbose posting for post-mission analysis of logic applied during any tag request event

5.7 The Results of a Tag Request (Tag Manager Book Keeping)

After a successfully applied tag, a few things happen internally to the tag manager worth noting. First, the tag time is associated with the source vehicle. In a subsequent tag request, the time of the previous tag is used for determining if the minimum time interval has been respected. This minimum interval is configured in the `tag_min_interval` parameter.

Second, the tag time is noted associate with the target name. A tag will expire after a certain period, determined by the `tag_duration` parameter. On each iteration of the tag manager, all vehicles currently in a tagged state are checked to see if their tag has expired. Third, the state of tagged vehicle is noted so that subsequent tag requests by the other team are not applied to an already-tagged vehicle.

5.8 Handling Tag Expiration

Once a vehicle has been tagged, the tag will expire after a set amount of time. This duration is configured in the `tag_duration` parameter, and by default is 30 seconds. A release event will be accompanied by a posting containing the released/untagged vehicle name and time at which it was released:

```
TAG_RELEASE_VERBOSE = "vname=henry,time=234.1"
```

5.9 Handling Tag and UnTag Postings

The tag manager may be configured to post arbitrary MOOS variable-value pairs whenever a tag has been applied or released. This is where the *effect* of a tag is determined, e.g., forcing a vehicle to stop or return, or slow down. Likewise the tag release may be configured to allow a vehicle to resume moving or speed up and so on. Posts made up on a tag are configured with the `tag_post` parameter. For example in the example mission:

```
tag_post = MOOS_MANUAL_OVERRIDE_$UP_TARGET = true
```

In the above example, a tag event where "abe" is tagged, will publish `MOOS_MANUAL_OVERRIDE_ABE=true`. In this mission, this will result in the halting of the tagged vehicle. The `tag_post` parameter allows for the expansion of the below macros:

- `$TARGET`: Expands to the name of the vehicle being tagged.
- `$UP_TARGET`: Expands to the upper case name of the vehicle being tagged
- `$SOURCE`: Expands to the name of the vehicle applying the tag.
- `$UP_SOURCE`: Expands to the upper case name of the vehicle applying the tag.
- `$TIME`: Expands to the local time of the event, i.e., the number of seconds since the start of the MOOSDB to which the tag manager is connected.
- `$UTC_TIME`: Expands to the UTC time of the event

The target and source macros will be expanded if used in both the variable name and variable value. The time macros are only expanded if used in the variable value.

When a tag expires, resulting in a release or untag event, postings may also be configured with the untag_post configuration parameter. The same macro scheme holds as in the case configuring tag posts, except the source macros are not supported since there is no notion of a source vehicle in an untag event.

6 Terminal and AppCast Output

The uFldTagManager application produces some useful information to the terminal on every iteration of the application. An example is shown in Listing 2 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 uFldTagManager, and serves a bit as a heartbeat indicator. The "0/0" also on line 2 indicates there are no configuration or run warnings detected.

The output in the below example comes from the example described in Section 7.

Listing 6.2: Example terminal or appcast output for uFldTagManager.

```
1  ================================================================
2  uFldTagManager alpha                                      0/0(834)
3  ================================================================
4  Global Settings
5  ===========================
6  Tag Range:    50
7  Tag Interval: 2
8  Team [blue]: betty (1)
9  Team [red]: archie (1)
10
11 Tag Application Stats:
12 ================================================
13         ReQ   Rejec  Rejec            Applied  Time
14 Name    Tags  Zone   Freq   Accepted  Tags     Next
15 ------  ----  -----  -----  --------  -------  ----
16 archie  2     1      0      1         1        n/a
17 betty   0     0      0      0         0        n/a
18
19 Tag Receiver Stats:
20 ================================================
21         Times   Currently  Time
22 Name    Tagged  Tagged     Remain  Taggable
23 ------  ------  ---------  ------  --------
24 betty   1       false      0.00    true
25 ------  ------  ---------  ------  --------
26 archie  0       false      0.00    true
27
28 ================================================================
29 Most Recent Events (4):
30 ================================================================
31 [115.59]: event=2,src=archie,team=redrejected=zone
32 [115.59]: Tag requested by archie[2]
33 [54.27]: event=1,src=archie,team=redtagged=betty
34 [54.27]: Tag requested by archie[1]
```

The first few lines (4-11) show the configuration settings for uFldTagManager. The status of uFldTagManager is shown in Lines 13-26.

7 A Simple Example

The s1_alpha_utmgr example mission distributed with moos-ivp-aquaticus tree provides a simple working example. More explanation to come...