# uFldHazardMetric: Grading a HazardSet Report

**June 2018**

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

**1 Overview** **1**

**2 Using uFldHazardMetric** **2**
2.1 Required MOOS Variables Shared between MOOSDBs . . . 3
2.2 The False-Alarm and Missed-Hazard Reward Structure . . . 4
2.3 The Max-Time and Time-Overage Reward Structure . . . 4
2.4 Raw and Normalized Scores . . . 5
2.5 The Report Evaluation Format . . . 5
2.6 Publishing the Mission Evaluation Parameters . . . 6

**3 Configuration Parameters of uFldHazardMetric** **6**

**4 Publications and Subscriptions for uFldHazardMetric** **7**
4.1 Variables Published by uFldHazardMetric . . . 7
4.2 Variables Subscribed for by uFldHazardMetric . . . 8

**5 Terminal and AppCast Output** **8**

**6 The Jake Kasper Example Mission Using uFldHazardMetric** **10**

## 1 Overview

The `uFldHazardMetric` application is a utility for quickly evaluating a hazardset report; a list of declared hazards and their locations. Evaluating a hazardset report against ground truth and a reward structure is fairly straight-forward, but tedious. This tool performs this operation automatically, and as a MOOS process with the result posted both to the MOOSDB and viewable in the appcast output of `uFldHazardMetric`. Operation is comprised of a few simple parts:

1. *Import a ground-truth hazard field*: A ground truth hazard field is a text file listing the location of hazards and hazard-like objects, and their locations. This file also typically includes a search region, a convex polygon containing all listed objects. A `uFldHazardMetric` configuration parameter names the file.
2. *Import a reward structure*: A reward structure, consisting of penalties for missed hazards and false alarms, is imported as a set of `uFldHazardMetric` configuration parameter.
3. *Evaluate a hazardset report*: A hazardset report is received by MOOS mail and evaluated item by item against the ground truth and reward structure. The results are then posted and rendered. This step is repeated for each received report.

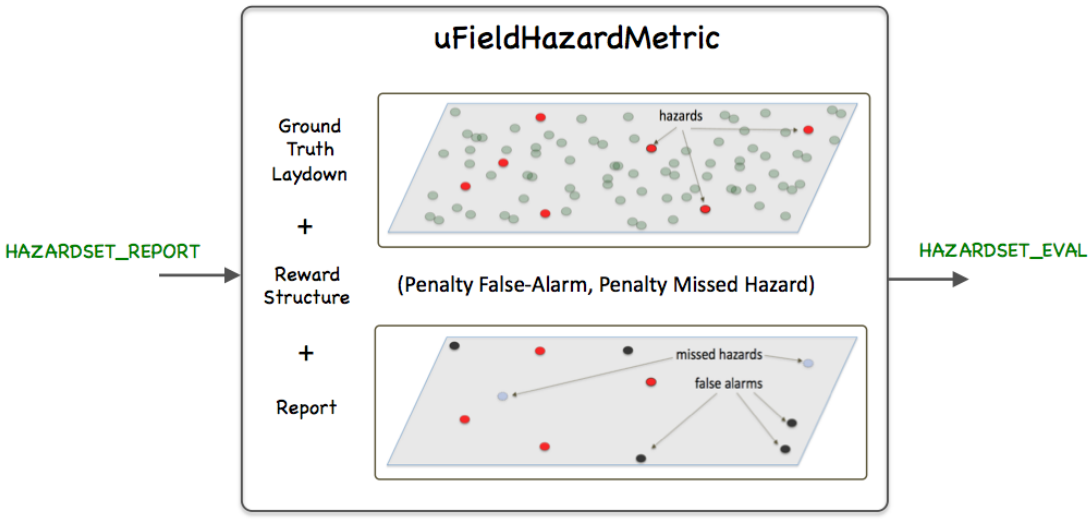


Figure 1: **The uFldHazardMetric**: interacts with the on-board sensor and processes sensor information and generates a report, upon request, regarding the identification and location of hazards. The arrows indicate the key MOOS variables used for interacting with the sensor and generating reports.

## 2 Using uFldHazardMetric

Typical use of `uFldHazardMetric` has it situated in the shoreside community, with hazardset reports shared from vehicles to the shoreside, and evaluation results shown via the `uFldHazardMetric` appcast output running on `pMarineViewer`. Full evaluation reports are also logged to the shoreside log file for later reference. This usage scenario with variations is described next. An example of this usage is in the Jake Kasper example mission described in Section 6.

### Typical Module Topology

The typical module topology is shown in Figure 2 below. The `uFldHazardMetric` is situated in the shoreside MOOS community. It does *not* interact with the `uFldHazardSensor` directly, but they are typically both configured with the same ground truth hazard file. `HAZARDSET_REPORT` messages are assumed to come from the vehicle. In the usage case below, they are produced by `uFldHazardMgr`. But as the latter is simply a straw-man sensor processing module, that could be replaced with something else entirely. `HAZARDSET_REPORT_EVAL` messages may be shared back to the vehicle, but this is likely not essential, as the typical destination of an evaluation is the appcast output and the log file.

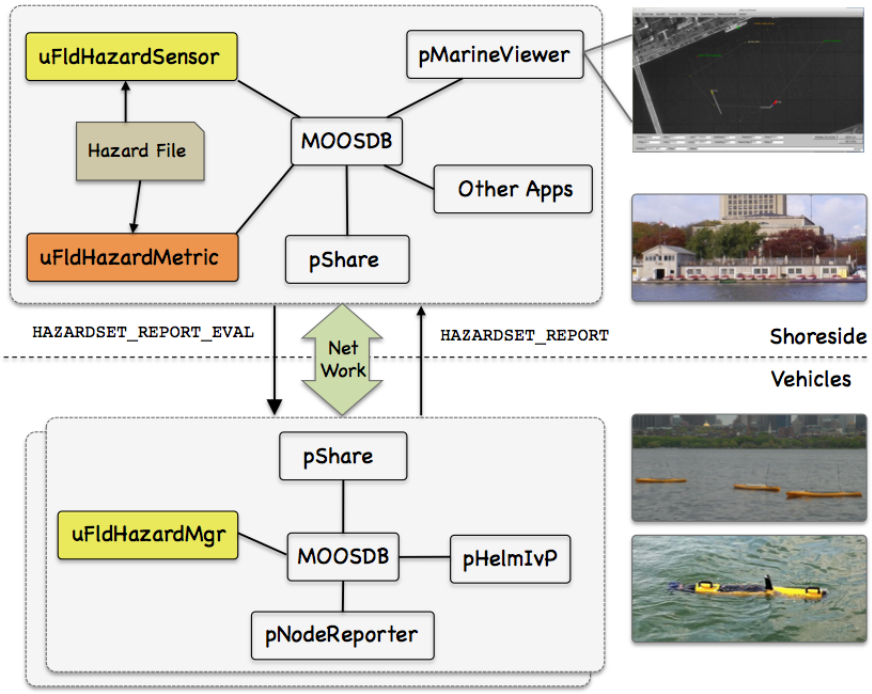


Figure 2: **Typical uFldHazardMetric Topology:** This module runs on the shoreside, alongside the hazard sensor typically, and receives hazazardset reports from the vehicle shared with the HAZARDSET_REPORT variable. Evaluations may be seen via the appcast output of uFldHazardMetric, or in the log file.

## 2.1 Required MOOS Variables Shared between MOOSDBs

Using uFldHazardMetric requires certain information flowing between the shoreside and vehicle communities as shown in Figure 2. Sharing is done by pShare, but the pShare configuration is handled dynamically using the uFldNodeBroker and uFldShoreBroker applications. We discuss here the necessary configuration entries for these two applications. From the vehicle to the shoreside, one variable needs to be shared. The below line should appear in the uFldNodeBroker configuration block on all vehicles.

```
bridge =  src=HAZARDSET_REPORT      // in uFldNodeBroker config block
```

This HAZARDSET_REPORT variable constitutes the report generated by uFldHazardMetric or a similar module running in the vehicle generating a hazardset report. The line may also be found in the vehicle configuration for the Jake Kasper example mission discussed in Section 6.

Going in the other direction, from shoreside to vehicle, the below line should appear in the uFldShoreBroker configuration block in the shoreside MOOS community. See the documentation for uFldShoreBroker for a discussion on the syntax.

```
// Bridge from Shoreside to Vehicle - in uFldShore Broker configuration
bridge =  src=HAZARDSET_REPORT_EVAL_$V, alias HAZARDSET_REPORT_EVAL
```

It may not be the case that your vehicle is actually utilizing the report evaluation, so the above may be optional. And certainly a typical mission will need to share other variables besides these, but from the perspective of uFldHazardMetric, these are the shares to make sure are configured. The above couple lines may also be found in the shoreside configuration for the Jake Kasper example mission discussed in Section 6.

### 2.2 The False-Alarm and Missed-Hazard Reward Structure

The primary metric for evaluating a hazardset report is based on penalties assigned to missed hazards and false alarms. The penalties are set with the parameters:

```
penalty_missed_hazard = <number>    // The default is 100
penalty_false_alarm   = <number>    // The default is 10
```

With $k_1$ missed hazards and $k_2$ false alarms, the penalty is:

$$penalty(k_1, k_2) = penalty_{MH}(k_1) + penalty_{FA}(k_2)$$

### 2.3 The Max-Time and Time-Overage Reward Structure

An optional additional metric may be applied which penalizes the report if it is late, with additional potential penalties the longer it is late. The time penalties are set with following parameters, beginning with the max_time parameter setting the point when a report is considered *late*:

```
max_time              = <number>  // seconds, default is 0
penalty_max_time_over = <number>  // penalty units, default is 0
penalty_max_time_rate = <number>  // penalty units, default is 0
```

The penalty_max_time_over parameter indicates the immediate one-time penalty applied if the report is late at all. The penalty_max_time_overage penalty is applied for *each second* of time past the deadline. If max_time is zero, there is no mission time limit.

If $t$ is the amount of time over the max time:

$$penalty(k_1, k_2, t) = \begin{cases} penalty_{FA}(k_1) + penalty_{MH}(k_2) & t \leq 0 \\ penalty_{FA}(k_1) + penalty_{MH}(k_2) + penalty_{TO} + penalty_{TR}(t) & t > 0 \end{cases}$$

The search duration clock re-starts each time uFldHazardMetric receives incoming mail on the variable HAZARD_SEARCH_START, regardless of the variable's value. Typically this variable is posted upon vehicle deployment as is done in the Jake Kasper example mission. (Hint: see how button_one is configured for pMarineViewer in the Jake Kasper mission.)

### 2.4 Raw and Normalized Scores

Past experience has shown that people appreciate a normalized score. A goal of zero (no penalties, perfect score) is somehow not as motivating as striving for 100% on a scale of zero to 100. A normalized score is derived from considering the worst possible score if each object in the hazard file were reported wrong. The score may be worse than this if the report is late and there are late penalties, but time is not used for the purposes of normalizing.

If $j_1$ and $j_2$ are the actual number of hazards and benign objects taken from ground truth in the hazard file, the worst score (without applying overtime penalties) is:

$$maxpenalty(j_1, j_2) = penalty_{MH}(j_1) + penalty_{FA}(j_2)$$

The normalized score is then:

$$score(k_1, k_2, t) = \frac{maxpenalty(j_1, j_2) - penalty(k_1, k_2, t)}{maxpenalty(j_1, j_2)}$$

If $penalty(k_1, k_2, t)$ is actually greater than $maxpenalty(j_1, j_2)$ due to lateness, resulting in a negative score, the normalized score is clipped to zero.

### 2.5 The Report Evaluation Format

The evaluation of the hazardset report has two formats, a terse and and verbose form. The terse form, `HAZARDSET_EVAL`, fully explains the score, the metrics, and the components of the submitted report responsible for the score. It may looks something like the example below from the Jake Kasper example mission:

```
HAZARDSET_EVAL =  vname=jake,                 report_name=BillandJoe,
                  total_score=675,            norm_score=37.5,
                  score_missed_hazards=500,   score_false_alarms=175,
                  score_time_overage=0,       total_objects=10,
                  total_time=1284.91,         received_time=1314.05,
                  start_time=29.14,           missed_hazards=5,
                  correct_hazards=5,          false_alarms=5,
                  penalty_false_alarm=35,     penalty_missed_hazard=100,
                  penalty_max_time_over=100,  penalty_max_time_rate=0.05,
                  max_time=1800
```

The full evaluation, `HAZARDSET_EVAL_FULL` provides all the details about which hazards were declared and missed, and which benign objects were false alarms. It may look something like the example below from the Jake Kasper example mission:

```
HAZARDSET_EVAL_FULL = (Everything in the normal report),object_report={
                      label=01,truth=hazard,report=hazard#
                      label=02,truth=hazard,report=nothing,penalty=100#
                      label=03,truth=hazard,report=hazard#
                      label=04,truth=hazard,report=nothing,penalty=100#
```

```
...
label=15,truth=benign,report=hazard,penalty=35#
label=16,truth=benign,report=hazard,penalty=35#
label=17,truth=benign,report=nothing#
label=18,truth=benign,report=hazard,penalty=35}
```

The latter perhaps may be simply used for forensics, or perhaps if further clarity is needed in how a scoring was applied.

### 2.6 Publishing the Mission Evaluation Parameters

Upon startup, the mission parameters are published by uFldHazardMetric on the shoreside and sent to each of the vehicles. Presumably this is to allow designers of the behavior autonomy to automatically adapt their mission to metrics that may not be known until mission launch time. The format of this message may look something like:

```
UHZ_MISSION_PARAMS = penalty_missed_hazard=100,
                     penalty_false_alarm=35,
                     max_time=600,
                     penalty_max_time_over=200,
                     penalty_max_time_rate=0.45,
                     search_region = pts={-150,-75:-150,-50:40,-50:40,-75}
```

## 3 Configuration Parameters of uFldHazardMetric

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

*Listing 3.1: Configuration Parameters for* uFldHazardMetric.

| | |
|---|---|
| penalty_missed_hazard: | The penalty for a missed hazard. The default is 100. Section 2.2. |
| penalty_false_alarm: | The penalty for a false alarm. The default is 10. Section 2.2. |
| penalty_max_time_over: | The penalty for submitting a report late. The default is zero. Section 2.3. |
| penalty_max_time_overage: | The penalty for submitting a late report, applied to every second it is late. The default is zero. Section 2.3. |
| max_time: | The time after which a submitted report is considered late. The default is zero, indicating there is no time limit. Section 2.2. |
| hazard_file: | The name of a hazard file naming the ground truth hazard field. |

### An Example MOOS Configuration Block

To see an example MOOS configuration block, enter the following from the command-line:

```
$ uFldHazardMetric --example or -e
```

This will show the output shown in Listing 2 below.

*Listing 3.2: Example configuration of the* uFldHazardMetric *application.*

```
1  ============================================================
2  uFldHazardMetric Example MOOS Configuration
3  ============================================================
4
5  ProcessConfig = uFldHazardMetric
6  {
7    AppTick   = 4
8    CommsTick = 4
9
10   penalty_missed_hazard = 100   // default
11   penalty_false_alarm   = 10    // default
12   penalty_max_time_over = 0     // default
13   penalty_max_time_rate = 0     // default
14
15   max_time      = 0             // default (no time limit)
16   hazard_file   = hazards.txt
17 }
```

# 4 Publications and Subscriptions for uFldHazardMetric

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

```
$ uFldHazardMetric --interface or -i
```

## 4.1 Variables Published by uFldHazardMetric

- 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 5
- HAZARDSET_EVAL: The shorter version of a hazardset report evaluation. Section 2.5.
- HAZARDSET_EVAL_FULL: The longer version of a hazardset report evaluation. Section 2.5.
- HAZARDSET_EVAL_<VNAME>: The shorter version of a hazardset report evaluation. The vehicle from which the report was received is appended to the evaluation so it may be shared only back to that vehicle. Section 2.5.
- HAZARDSET_EVAL_<VNAME>: The longer version of a hazardset report evaluation. The vehicle from which the report was received is appended to the evaluation so it may be shared only back to that vehicle. Section 2.5.
- HAZARD_SEARCH_SCORE: The normalized score reported in HAZARDSET_EVAL published as a single numerical value.
- UHZ_MISSION_PARAMS: A list of the mission parameters used for scoring a submitted HAZARD_REPORT.
- VIEW_POLYGON: A polygon rendering the search area as defined in the hazard file.

### 4.2 Variables Subscribed for by uFldHazardMetric

The uFldHazardMetric 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.
- HAZARDSET_REPORT: An incoming hazardset report. See the uFldHazardSensor documentation.
- HAZARD_SEARCH_START: An indication that the clock used to apply time limits and penalties is to be restarted. Section 2.3.

### Command Line Usage of uFldHazardMetric

The uFldHazardMetric application is typically launched as a part of a batch of processes by pAntler, but may also be launched from the command line by the user. To see command-line options enter the following from the command-line:

```
$ uFldHazardMetric --help or -h
```

This will show the output shown in Listing 3 below.

*Listing 4.3: Command line usage for uFldHazardMetric.*

```
1  ============================================================
2  Usage: uFldHazardMetric file.moos [OPTIONS]
3  ============================================================
4
5  Options:
6    --alias=<ProcessName>
7        Launch uFldHazardMetric with the given process name.
8    --example, -e
9        Display example MOOS configuration block.
10   --help, -h
11       Display this help message.
12   --interface, -i
13       Display MOOS publications and subscriptions.
14   --version,-v
15       Display release version of uFldHazardMetric.
```

## 5 Terminal and AppCast Output

The uFldHazardMetric application produces some useful information to the terminal and identical content through appcasting. An example is shown in Listing 4 below. On line 2, the name of the local community, typically the shoreside community, is listed on the left. On the right, "0/0(5429) indicates there are no configuration or run warnings, and the current iteration of uFldHazardMetric is 5429. Lines 4-6 show the name of the ground truth hazard file and the number of hazards and benign objects. Lines 8-13 convey the requested and prevailing configuration settings for evaluating incoming reports.

*Listing 5.4: Example uFldHazardMetric console output.*

```
1  ================================================================
```

```
 2  uFldHazardMetric shoreside                                    0/0(5429)
 3  ================================================================
 4  Hazard File: (hazards.txt)
 5       Hazard: 10
 6       Benign: 8
 7
 8  Reward Structure:
 9      Penalty Missed Hazard: 100
10        Penalty False Alarm: 35
11      Penalty Max Time Over: 100
12      Penalty Max Time Rate: 0.05
13                    Max Time: 1800
14
15  ========================================
16  Received Reports: 6
17  Elapsed Time:     1285.76
18  ========================================
19  Report     Total    Time      Time     Raw    Norm
20  Name       Reports  Received  Elapsed  Score  Score
21  --------   -------  --------  -------  -----  -----
22  Sarah      6        1351.97   1278.28  675    37.5
23
24  ========================================
25  Most Recent Report: (jake/Sarah)
26            total_score: 675 (37.5)
27   score_missed_hazards: 500 (5)
28     score_false_alarms: 175 (5)
29     score_time_overage: 0 (0)
30  --------------------
31       objects reported: 10
32        correct_hazards: 5 (of 10)
33
34  ================================================================
35  Most Recent Events (7):
36  ================================================================
37  [1351.97]: Received valid report from: jake
38  [1347.46]: Received valid report from: jake
39  [1181.93]: Received valid report from: jake
40  [1178.42]: Received valid report from: jake
41  [641.28]: Received valid report from: jake
42  [631.76]: Received valid report from: jake
43  [0.00]: Reading hazards.txt: Objects read: 18
```

Lines 15-22 provide a summary of reports received so far, perhaps from multiple vehicles. Line 16 shows the number of received reports total from *all* vehicles, and line 17 shows the elapsed time since the receipt of `HAZARD_SEARCH_START` as discussed in Section 2.3. Beginning with line 19, for each vehicle the total reports received are shown, plus information about the last received report from that vehicle. The last four columns show (a) the time the report was received, (b) the elapsed time for the report since the latest timer reset, (c) the raw score as discussed in Sections 2.2 and 2.3, and (d) the normalized score as discussed in Section 2.4.

Lines 24-32 dive into more detail about the latest report received from any vehicle. Line 25 shows the name of the report which may consist of both a vehicle name and additional report name. The total raw and normalized score is shown next on line 26, with the justification for the score

shown next in lines 27-32. Following this block of output, events are shown starting here on line 34. In this case most events simply report the arrival of new reports, but other events may indicate anomalous activities such as the arrival of an empty report.

# 6 The Jake Kasper Example Mission Using uFldHazardMetric

The *Jake Kasper* mission is distributed with the MOOS-IvP source code and contains a ready example of the `uFldHazardMetric` application, configured with a hazard field in an included text file. Assuming the reader has downloaded the source code available at `www.moos-ivp.org` and successfully built the code, the *Jake Kasper* mission may be launched by:

```
$ cd moos-ivp/ivp/missions-2680/lab_10_jake_kasper_baseline/
$ ./launch.sh 12
```

The argument, 12, in the line above will launch the simulation in `12x` real time. Once this launches, the `pMarineViewer` GUI application should launch and the mission may be initiated by hitting the `DEPLOY_ALL` button. Shortly thereafter, two vehicles named `jake` and `kasper` will begin simple lawnmower patterns over half of the search region each, as shown in Figure 3 below.

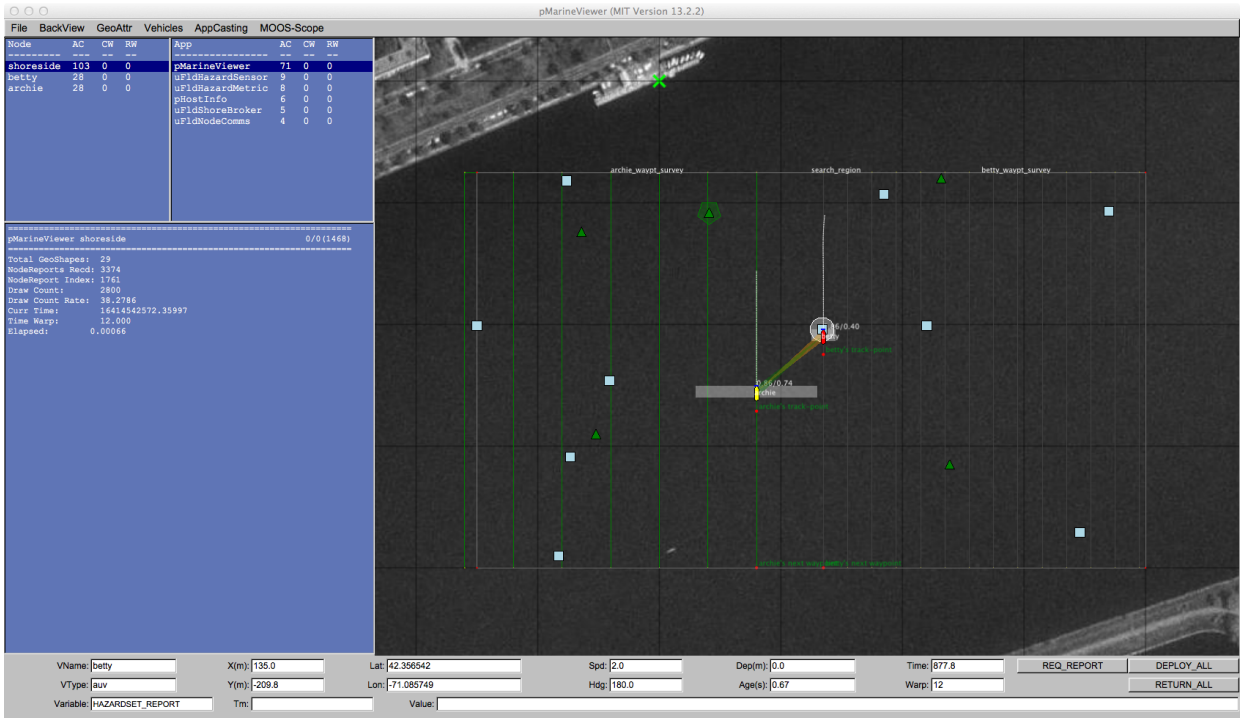


Figure 3: **uFldHazardMetric in the Jake Kasper example mission**: The output of uFldHazardMetric is shown in the appcast panel in the lower left after a hazardset report has been received and evaluated.

Any time after the vehicle has been deployed, the user may request the generation of a hazardset

report. The REQ_REPORT button sends a HAZARDSET_REQUEST message to the vehicles, each running uFldHazardMgr. Repeated requests result in updated reports. The overall score of the reports tends higher as the mission progresses and more hazards are detected.