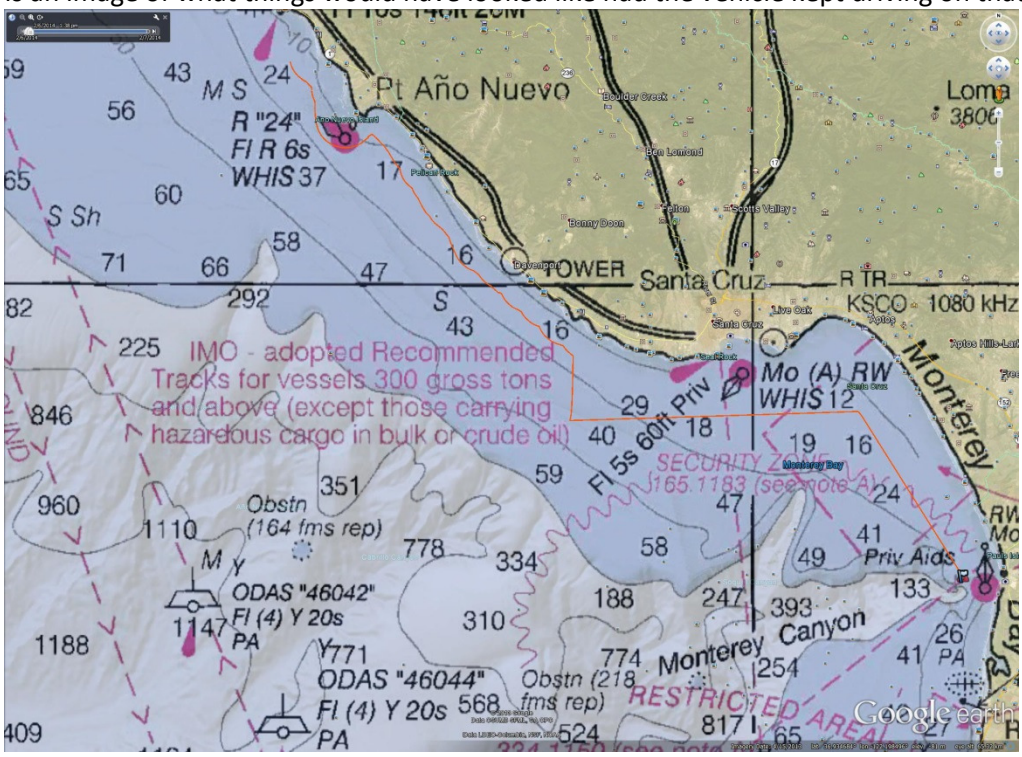
Description:

Two main problems contributed to the unplanned *Tethys* recovery that occurred on 2014/02/09.

First, the easy one: Why was the vehicle so far North? The reason was a copy/paste error that sent the following command “set science_to.Wpt3Lon 0-122.130”

The vehicle decided to turn that command into this: “2014-02-05T02:11:08.306Z,1391566268.306 [CommandLine](IMPORTANT): got command set science_to.Wpt3Lon 122.129997 degree”
That location is actually in the Yellow Sea just off China. Offshore envelope worked well and kept us off the rocks. Below is an image of what things would have looked like had the vehicle kept driving on that command.

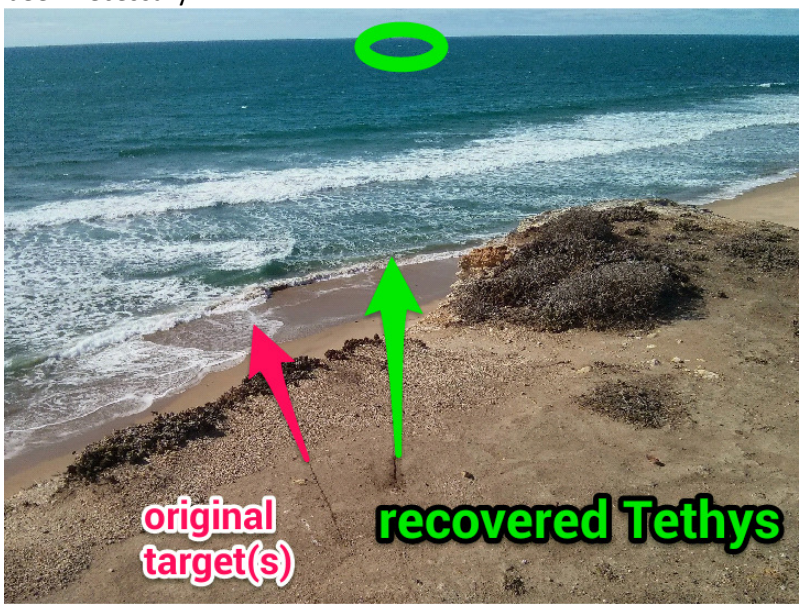


Second, why did the vehicle stop responding?
In short, the cause is due to the non-atomic nature of a universal data writer’s validity and accuracy being changed. The reader might get a return value of true when asking for validity, but an invalid accuracy value one line later. This failure is a complex interaction between the asynchronous DVL component and the main thread that caused an occasional race condition. It took multiple failures during a single dive with subtle timing interactions to cause the problem to occur. Details follow below.

Tethys was found WNW of Sand Hill Bluff about 1.5 km from the beach and recovered on the Vessel Assist boat *Jeanie* in the early afternoon on February 9, 2014. Upon recovery the elevators were in the full climb position and the prop was off. No communication channels were active. Upon further inspection, lat/lon were being written as not a number (nan), and depth was consequently being reported as nan despite good pressure data being recorded. This prevented the radios from being activated when on the surface.
The depth interface already contains a safety check to make sure the vehicle has a valid location. If it doesn’t, a best estimate from the configuration will be used. In this particular case, the catch didn’t work since it’s based on latitude

being active and providing a valid read of the data variable. Both occurred in this case, but the value was nan. Applying a pressure calculation to a nan results in…. a nan.

Notes: RDF was very handy for locating a disabled vehicle on the surface from the cliffs and from the boat. It would also work well from the air. Ignoring drift, the vehicle was recovered about 20 degrees off of the original bearing sited via the RDF approximately two hours prior. Recovery on Argos/visual alone would have been very difficult given the wind chop and would have likely split our resources between beach searches and water searches. Acoustic tracking would have been necessary.



Details on nan being written for location:

This section describes why the aforementioned nan was being written for vehicle location. Slate.asc snippets are taken from the file slate_1330_slate_1530_201402051330_201402051529.asc and code snippets from the tag Tethys_2014_01_31. And now, the details:

The DVL interface runs asynchronously and, upon receiving invalid data, will set its writers invalid. See the following example:

Ex1:

2014-02-05T13:49:52.497Z,1391608192.497 [DVL_micro](set state of DVL_micro.platform_speed_wrt_ground): invalid
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of Unknown): best
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of DVL_micro.platform_x_velocity_wrt_ground): invalid
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of Unknown): best
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of DVL_micro.platform_y_velocity_wrt_ground): invalid
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of Unknown): best
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of DVL_micro.platform_z_velocity_wrt_ground): invalid
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of Unknown): best
2014-02-05T13:49:52.498Z,1391608192.498 DVL_micro-->DVL_micro.BottomVelocityFlag=0 count
2014-02-05T13:49:52.498Z,1391608192.498 [DVL_micro](set state of DVL_micro.platform_velocity_wrt_ground): invalid

During this time other components, like navigators, can be running and querying for a particular writer's validity. In this case, we see DeadReckonUsingMultipleVelocitySources falling into the following if statement despite the fact that the writers were just set invalid.

Ex2:

2014-02-05T13:49:52.505Z,1391608192.505 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.velocitySource=0 count
2014-02-05T13:49:52.505Z,1391608192.505 [DeadReckonUsingMultipleVelocitySources](**set accuracy of DeadReckonUsingMultipleVelocitySources.latitude): 330565653800875164958720.00000**
2014-02-05T13:49:52.505Z,1391608192.505 [DeadReckonUsingMultipleVelocitySources](set state of DeadReckonWithRespectToWater.latitude): best
2014-02-05T13:49:52.505Z,1391608192.505 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.latitude=36.963543 arcdeg

2014-02-05T13:49:52.506Z,1391608192.506 [DeadReckonUsingMultipleVelocitySources](set accuracy of DeadReckonUsingMultipleVelocitySources.longitude): 330565653800875164958720.00000
2014-02-05T13:49:52.506Z,1391608192.506 [DeadReckonUsingMultipleVelocitySources](set state of DeadReckonWithRespectToWater.longitude): best
2014-02-05T13:49:52.506Z,1391608192.506 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.longitude=-122.169878 arcdeg
2014-02-05T13:49:52.506Z,1391608192.506 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.depth=19.599121 m
2014-02-05T13:49:52.506Z,1391608192.506 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.durationOfLastRun=0.010376 s
2014-02-05T13:49:52.507Z,1391608192.507 DeadReckonWithRespectToWater==>DeadReckonWithRespectToWater.latitude=36.964092 arcdeg
2014-02-05T13:49:52.507Z,1391608192.507 DeadReckonWithRespectToWater==>DeadReckonWithRespectToWater.longitude=-122.171693 arcdeg

Ex3:

```
if( !( uBottomReader_->isInvalid() ) && !( vBottomReader_->isInvalid() ) && !( wBottomReader_->isInvalid() ) )
{
 …
   velocityRelativeToGroundInVehicleFrame_.setU( uBottomReader_->asDouble( Units::METER_PER_SECOND ) );
   velocityRelativeToGroundInVehicleFrame_.setV( vBottomReader_->asDouble( Units::METER_PER_SECOND ) );
   velocityRelativeToGroundInVehicleFrame_.setW( wBottomReader_->asDouble( Units::METER_PER_SECOND ) );
   speed_ = velocityRelativeToGroundInVehicleFrame_.sumSquared();
   speedAccuracy_ = uBottomReader_->getAccuracy( Units::METER_PER_SECOND );
   velocitySourceWriter_->write( Units::COUNT, 0 );
 …
```

The 0.002 seconds was not enough time for the isInvalid() call to return true and the navigator will not set a nan for its accuracies. In this case, a large accuracy called NO_ACCURACY is set as 3.25e30. This number, after being multiplied by the inverse of earth's radius, ends up being the ~3e24 number shown above. Also note that the best lat/lon writer has now switched over to DeadReckonWithRespectToWater due to this accuracy change. Depending on the race condition, DeadReckonUsingMultipleVelocitySources may write a nan to the non-best (i.e. non universal) with the very bad accuracy. This will come into play later. Even after changing all navigators to test validity of data based on the return value of the read function, this accuracy problem can still occur.

So now the vehicle sits in a state where DeadReckonUsingMultipleVelocitySources should be the best writer for lat/lon as water velocity is still good. velocitySource should have been written as a 1 and the velocity passed to the navigator should have been water velocity. If the vehicle remains underwater in this state there will be no GPS fix to reduce the accuracy back to a reasonable number for DeadReckonUsingMultipleVelocitySouces. Thus, DeadReckonWithRespectToWater will remain "best".

Further on, we see another instance of this race condition. DVL sets the writer invalid, but DeadReckonUsingMultipleVelocitySources is still writing a 0 for velocitySource meaning the writers were valid at the time of checking (see if statement Ex3). The big difference in this case is that the value returned from asDouble was a nan which gets written to the non-universal lat/lon (below) with an accuracy that's really big but still not NAN. Note that in the previous instance, Ex2, the value returned by asDouble() was valid.

Ex4:

2014-02-05T14:20:11.881Z,1391610011.881 [DVL_micro](set state of DVL_micro.platform_speed_wrt_ground): invalid
2014-02-05T14:20:11.881Z,1391610011.881 [DVL_micro](set state of Unknown): best
2014-02-05T14:20:11.881Z,1391610011.881 [DVL_micro](set state of DVL_micro.platform_x_velocity_wrt_ground): invalid
2014-02-05T14:20:11.881Z,1391610011.881 [DVL_micro](set state of Unknown): best
2014-02-05T14:20:11.881Z,1391610011.881 [DVL_micro](set state of DVL_micro.platform_y_velocity_wrt_ground): invalid
2014-02-05T14:20:11.882Z,1391610011.882 [DVL_micro](set state of Unknown): best
2014-02-05T14:20:11.882Z,1391610011.882 [DVL_micro](set state of DVL_micro.platform_z_velocity_wrt_ground): invalid
2014-02-05T14:20:11.882Z,1391610011.882 [DVL_micro](set state of Unknown): best
2014-02-05T14:20:11.882Z,1391610011.882 DVL_micro-->DVL_micro.BottomVelocityFlag=0 count
2014-02-05T14:20:11.882Z,1391610011.882 [DVL_micro](set state of DVL_micro.platform_velocity_wrt_ground): invalid
2014-02-05T14:20:11.890Z,1391610011.890 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.velocitySource=0 count
2014-02-05T14:20:11.890Z,1391610011.890 [DeadReckonUsingMultipleVelocitySources](set accuracy of DeadReckonUsingMultipleVelocitySources.latitude): 6564089411188806847037 44.00000
…
2014-02-05T14:20:11.894Z,1391610011.894 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.latitude=nan arcdeg
2014-02-05T14:20:11.895Z,1391610011.895 [DeadReckonUsingMultipleVelocitySources](set accuracy of DeadReckonUsingMultipleVelocitySources.longitude): 6564089411188806847037 44.00000
2014-02-05T14:20:11.895Z,1391610011.895 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.longitude=nan arcdeg
2014-02-05T14:20:11.895Z,1391610011.895 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.depth=nan m
2014-02-05T14:20:11.895Z,1391610011.895 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.durationOfLastRun=0.025391 s

2014-02-05T14:20:11.896Z,1391610011.896 DeadReckonWithRespectToWater==>DeadReckonWithRespectToWater.latitude=36.972862 arcdeg
2014-02-05T14:20:11.896Z,1391610011.896 DeadReckonWithRespectToWater==>DeadReckonWithRespectToWater.longitude=-122.181872 arcdeg
2014-02-05T14:20:11.897Z,1391610011.897 DeadReckonWithRespectToWater-->DeadReckonWithRespectToWater.depth=41.615234 m

Then as long as the vehicle is still underwater and hasn't gotten a GPS fix all it takes is an invalid DVL ping to cause DeadReckonWithRespectToWater to relinquish the writing to the next best accuracy. Since DeadReckonUsingMultipleVelocitySouces still has a valid, though very poor accuracy, it's next in line for the universal value of latitude and longitude. Due to the aforementioned timing issue, it has been writing nans for lat/lon. The last two bolded lines of this snippet show the nan being written to the universal denoted by the "==>". Once a nan gets in the navigator's calculation of position, it stays until a fix is received.

Contrast this with normal operation, outside of this timing issue that has been described, where each navigator sets its own accuracy to nan if it does not have valid velocity inputs. This prevents ever writing a nan to the slate.

Ex5:

2880801:2014-02-05T15:05:22.854Z,1391612722.854 [DVL_micro](ERROR): DVL Failure: Failed to read in all data items. Got 29 of 46
2880802:2014-02-05T15:05:22.855Z,1391612722.855 [DVL_micro](ERROR): Failed to parse DVL response:$#NQ.RES 0X0000 1 1 1 1 14.2 10.8 10.0 12.4 -4.8 164.4 -24.5 -329 -62 394 -439 2 2 1 1 -653.3 -40.4 -64.8 1 -604.2 184.6 183.1 1 --117 2 -716 819 -117 2 6.60 22.24 338.7 10.0 12.5 1079507.968 35.0 1489 88
…
2880840:2014-02-05T15:05:22.894Z,1391612722.894 [DVL_micro](set state of DVL_micro.platform_speed_wrt_ground): invalid
2880841:2014-02-05T15:05:22.894Z,1391612722.894 [DVL_micro](set state of Unknown): best
2880842:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of DVL_micro.platform_x_velocity_wrt_ground): invalid
2880843:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of Unknown): best
2880844:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of DVL_micro.platform_y_velocity_wrt_ground): invalid
2880845:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of Unknown): best
2880846:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of DVL_micro.platform_z_velocity_wrt_ground): invalid
2880847:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of Unknown): best
2880848:2014-02-05T15:05:22.895Z,1391612722.895 DVL_micro-->DVL_micro.BottomVelocityFlag=0 count
2880849:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of DVL_micro.platform_velocity_wrt_ground): invalid
2880850:2014-02-05T15:05:22.895Z,1391612722.895 [DVL_micro](set state of DVL_micro.platform_speed_wrt_sea_water): invalid
2880851:2014-02-05T15:05:22.899Z,1391612722.899 HFRCMSurfaceCurrentAtVehicleLocation--
>HFRCMSurfaceCurrentAtVehicleLocation.surface_eastward_sea_water_velocity=-0.045551 m/s
2880852:2014-02-05T15:05:22.899Z,1391612722.899 HFRCMSurfaceCurrentAtVehicleLocation--
>HFRCMSurfaceCurrentAtVehicleLocation.surface_northward_sea_water_velocity=-0.120632 m/s
2880853:2014-02-05T15:05:22.899Z,1391612722.899 HFRCMSurfaceCurrentAtVehicleLocation--
>HFRCMSurfaceCurrentAtVehicleLocation.durationOfLastRun=0.009399 s
2880854:2014-02-05T15:05:22.901Z,1391612722.901 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.velocitySource=1 count
2880855:2014-02-05T15:05:22.902Z,1391612722.902 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.latitude=nan arcdeg
2880856:2014-02-05T15:05:22.902Z,1391612722.902 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.longitude=nan arcdeg
2880857:2014-02-05T15:05:22.902Z,1391612722.902 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.depth=nan m
2880858:2014-02-05T15:05:22.902Z,1391612722.902 DeadReckonUsingMultipleVelocitySources--
>DeadReckonUsingMultipleVelocitySources.durationOfLastRun=0.002090 s
2880870:2014-02-05T15:05:22.915Z,1391612722.915 [DVL_micro](set state of SpeedCalculator.platform_speed_wrt_sea_water): best
2880871:2014-02-05T15:05:22.915Z,1391612722.915 [DVL_micro](set state of DVL_micro.platform_x_velocity_wrt_sea_water): invalid
2880872:2014-02-05T15:05:22.915Z,1391612722.915 [DVL_micro](set state of Unknown): best
2880873:2014-02-05T15:05:22.915Z,1391612722.915 [DVL_micro](set state of DVL_micro.platform_y_velocity_wrt_sea_water): invalid
2880874:2014-02-05T15:05:22.915Z,1391612722.915 [DVL_micro](set state of Unknown): best
2880875:2014-02-05T15:05:22.915Z,1391612722.915 [DVL_micro](set state of DVL_micro.platform_z_velocity_wrt_sea_water): invalid
2880876:2014-02-05T15:05:22.915Z,1391612722.915 [DVL_micro](set state of Unknown): best
2880877:2014-02-05T15:05:22.916Z,1391612722.916 DVL_micro-->DVL_micro.WaterVelocityFlag=0.000000 m
2880878:2014-02-05T15:05:22.916Z,1391612722.916 [DVL_micro](set state of DVL_micro.platform_velocity_wrt_sea_water): invalid
2880896:2014-02-05T15:05:22.922Z,1391612722.922 [DeadReckonWithRespectToWater](DEBUG): Will not write estimated position: latitudeAccuracy_ = nan, longitudeAccuracy_ = nan, depthAccuracy_ = 1e+20
2880897:2014-02-05T15:05:22.922Z,1391612722.922 [DeadReckonWithRespectToWater](set state of DeadReckonWithRespectToWater.latitude): invalid
2880898:2014-02-05T15:05:22.922Z,1391612722.922 [DeadReckonWithRespectToWater](set state of DeadReckonUsingMultipleVelocitySources.latitude): best
2880899:2014-02-05T15:05:22.922Z,1391612722.922 [DeadReckonWithRespectToWater](set state of DeadReckonWithRespectToWater.longitude): invalid
2880900:2014-02-05T15:05:22.922Z,1391612722.922 [DeadReckonWithRespectToWater](set state of DeadReckonUsingMultipleVelocitySources.longitude): best
2880901:2014-02-05T15:05:22.922Z,1391612722.922 [DeadReckonWithRespectToWater](set state of DeadReckonWithRespectToWater.depth): invalid
…
2881272:2014-02-05T15:05:23.687Z,1391612723.687
DeadReckonUsingMultipleVelocitySources==>DeadReckonUsingMultipleVelocitySources.latitude=nan arcdeg
2881273:2014-02-05T15:05:23.687Z,1391612723.687
DeadReckonUsingMultipleVelocitySources==>DeadReckonUsingMultipleVelocitySources.longitude=nan arcdeg
2881274:2014-02-05T15:05:23.687Z,1391612723.687 DeadReckonUsingMultipleVelocitySources-->DeadReckonUsingMultipleVelocitySources.depth=nan m

It's worth noting that the race condition highlighted in Ex1-Ex3 is still present. isValid can return true, the read can return a valid value, and the accuracy can still be invalid. An atomic write of the value and accuracy associated with that value would be the only way to mitigate this issue.

Fixes:

- Depth interface uses best estimate when latitude is nan.
- Reader will check return value from read instead of isInvalid
- DVL interface needs to be re-written just for NQ1
- Navigator will read accuracy immediately before reading the value. This still leaves the possibility of the race condition highlighted above and cannot be mitigated without an architectural change to how component data with accuracy is written.
- Navigators include an additional check for nan in location just before writing. If a nan is found, the navigator logs the details to the syslog but does not write nans to the slate.