

# July 2007 Sea Trial

## Introduction

This document is presented in preparation for the coming sea trial (ST\_1) scheduled for July 24<sup>th</sup>. The primary goal of ST\_1 is to verify fixes and improvements since ST\_0. The following changes have been made to address issues from our first sea trial (ST\_0) in June:

- The **timing model** has been changed to handle a larger window of uncontrollability on the end times of commands. This increase is required because of observed worst-case bounds on latencies for message passing in the VCS. This fault accounted for a number of test case failures in ST\_0.
- The **check-in policy** has been changed to specify a minimum time at the surface rather than an exact time at the surface.
- The treatment of depth data for recognizing when **at the surface** or not. This was responsible for a number of test case failures in ST\_0. New policy:
  - If depth  $\leq 0.5$  then AtSurface = true.
  - If depth  $\geq 2.5$  then AtSurface = false.
  - Otherwise we do not know and do not commit.
- A new **logging mechanism** has been put in place to reduce the possibility of losing data as happened where we over-wrote some test output. This mechanism mirrors that of the VCS.
- **Performance improvements.** The cost for logging has been greatly reduced and a number of additional speed-ups have been added.

The secondary goal is to test **new functionality**. The focus here is:

- to move from single link missions to multi-link missions
- to incorporate science and operator goal driven commanding via the *Mission Manager*
- to run for longer

We will have about 6 hours on site to work with the vehicle. The plan will be to find a region of operation reasonably close to shore with a safe operating radius of about 1 km at the surface and 50 meters to the bottom. The tests are structured as follows:

- Safety Checks. Verify basic operation in nominal and off-nominal cases.
- Simple Mobility. Very primitive actions and commands for navigating.
- Check-in Tests. Very pre-emption and resumption capability to localize at the surface.
- Path Navigation. Demonstrate path-constrained navigation.
- Mission Management. From high-level goals to low-level details.

## Background

This section will briefly review some background material on the system.

### ***Features of Interest***

- **Vehicle Safety.** A key part of the system design is a robust fail-over when something goes wrong. The idea is to ensure the core Vehicle Control System can always resume control and that the vehicle must not be permitted to be out of active control for longer than a given period (i.e. 10 seconds).
- **Straight Plan Execution.** This is the ability of the agent to check and execute a given plan. It is useful to demonstrate this as plans can be constructed to reflect existing mission inputs. The agent in this manner dispatches a plan and monitors plan execution, but is not *generating new commands* (i.e. planning)
- **Reactive Planning.** This is where planning occurs *during synchronization* at the execution frontier. This allows the agent to quickly respond to feedback based on sensor values or command status transitions, and to refine active goals as execution unfolds.
- **Deliberative Planning.** This is where goals farther out than the execution frontier are considered and plans generated to accomplish them. It allows a longer horizon over which to generate a suitable plan and allows the agent to get prepared for what is to come.
- **Distributed Planning.** This is where the agent is planning over different temporal and functional scopes and they are synthesized and synchronized. It drives the full capabilities of TREX with respect to information flow in a multi-reactor configuration.

### ***Measurements of Interest***

- All log data from the VCS
- Log for Playback on the AMC
- AMC Resource Usage (CPU & Memory)
- Search Efficiency - Step Count and Node Count for Synchronization and Deliberation for all reactors.

### ***Behaviors***

In addition to the safety behaviors, the following behaviors are used:

- **Waypoint.** Attempts to achieve a position using a specified direction of approach. This behavior can be used on the surface (if traveling on the surface) or at depth.
- **Descend.** Drives the vehicle towards the bottom, to a specified depth. Terminates when a target depth or minAltitude are reached.
- **Ascend.** Drives the vehicle towards the surface. It terminates when the depth is less than a specified minDepth. We will use this to surface the vehicle.

- **GetGPS.** We will get the GPS whenever we reach the surface. It can only occur when at the surface. **It will be used with minHits = 30, and abortOnTimeout = true. It will have a max duration of 600 seconds.**
- **Waypoint\_Yoyo.** This behavior causes the vehicle to oscillate between two extremes of depth. Termination is based on a timeout or a completion of a specified number of cycles.
- **Setpoint.** This behavior sets heading, depth or pitch, and speed of the vehicle. This behavior is used to idle at the surface, to stop the prop, and to build speed for descent when at the surface.

## **Units**

- The unit of time will be a second. This is the finest granularity for all timing and also the units in which goals and constraints must be specified.
- All positional information will be provided in latitude and longitude. These are specified in degrees (rather than radians).
- All distances (horizontal and vertical) will be defined in meters.
- Speed will be in meters/per second

## **Safety**

The main strategy for safety is to fail over into an abort plan as soon as possible once a fault occurs, and to ensure the overall mission operates within a safety envelope for depth, altitude and time. This section reviews the design elements addressing vehicle safety.

## **Startup Plan**

The AMC ingests a base mission plan (init.cfg) in the standard AUV mission script format. We call this the *startup plan*. It contains the following:

- **Mission Timer.** This is a non-sequential behavior used to terminate the mission after the given duration has elapsed. It is defined in seconds. This allows the VCS to initiate termination of the mission. **The maximum mission duration we will permit is 30 minutes to insure we remain in our operating region. Most missions will be limited to 10.**
- **Depth Envelope.** This is a non-sequential behavior used to ensure the vehicle stays within a given depth range. We will have a **maximum depth of 50 meters and a minimum altitude of 10 meters.** The vehicle will be operating over a sandy bottom. In practice, missions will be commanded to no more than **30 meters depth.**
- **Setpoint.** This is the only sequential behavior in the base plan. It will be short (2 seconds). The completion of this behavior provides the **initialization signal** for the AMC.

## Watchdogs

Layered control has been modified to permit periods with no sequential behaviors on the stack. This is to allow some time for the AMC to generate behaviors reactively on termination of a behavior. This arises for example where the parameters for a waypoint are computed when the target depth is obtained. This may also arise if the plan breaks, triggering a recall, and requiring re-planning.

To enforce a maximum time for no behaviors on the stack, the AMC has been augmented with a watchdog timer that **starts when the command sequence timeline begins in an undefined state**. The timer is reset when the command sequence timeline switches to a defined command. **If the timer reaches a timeout of 10 seconds, it sends an exit command to the VCS. This exit immediately jumps into the abort plan.**

To capture conditions where the AMC is not working correctly from the VCS perspective, we have implemented a **heartbeat using a ping command generated from the AMC**. The AMC sends a ping every second under normal operation. If it crashes or locks up, no ping is sent. The VCS has a separate watchdog that is reset on receipt of each ping and will accumulate otherwise. **If it reaches an elapsed time of 10 seconds without being cleared, it will jump into the abort plan.**

## AMC Mission Timer

The AMC is configured with a mission duration upper bound. If the mission reaches this time, it terminates the mission. Termination will send an exit to the VCS, which will jump it into the abort plan. This will be set tightly on a case-by-case basis and will not exceed 30 minutes.

## Abort on Timeout

GetGPS is parameterized with `abortOnTimeout = true` in the model. We also check for post-conditions on ascend and descend thus detecting a plan failure. These issues are handled in the model.

## Stop Gaps

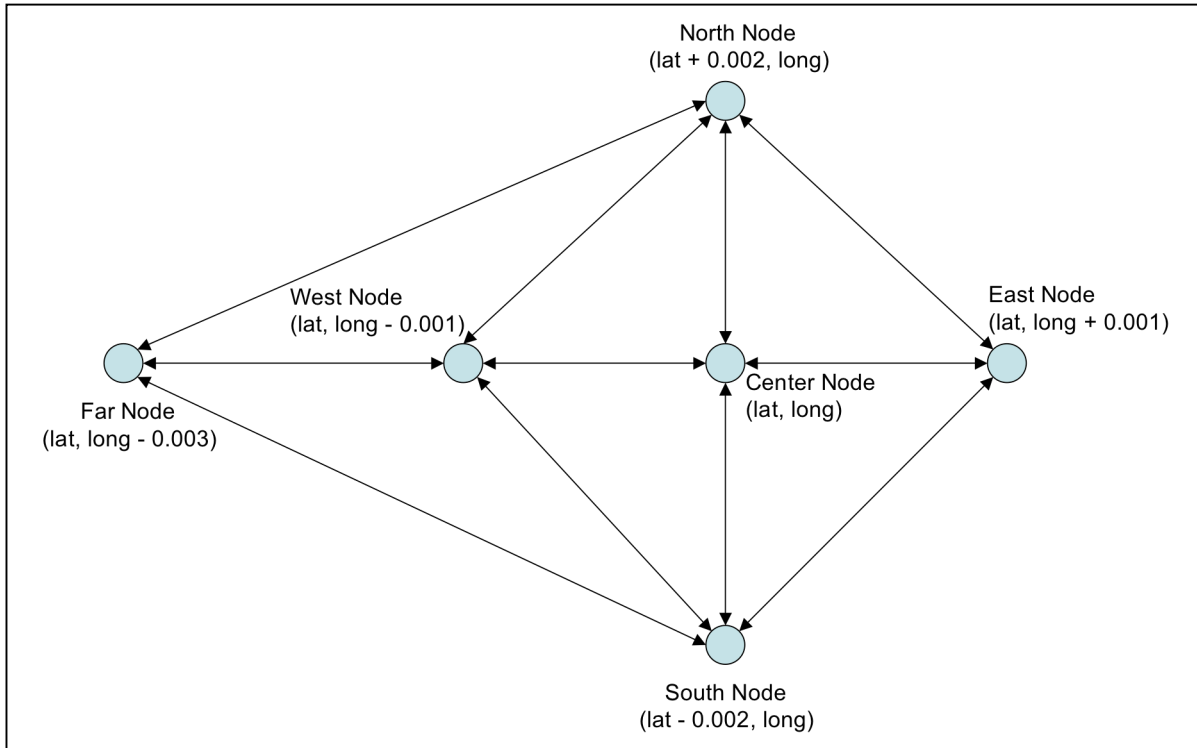
It may occur that a command ends and the stack is empty for a period of as much as 10 seconds (i.e. the watchdog timeout). The control parameters retain the values of the last command. **It will be up to the model to correctly ensure the prop is stopped if that is the desirable thing to do.**

## Abort Plan

Nominal completion of each mission will occur via execution of the abort plan. This will use a 0 speed setpoint with a duration of 90 minutes. It will then include a dropWeight.

## Region of Operation

The graph to be used for operation is shown below. All commanded destinations are relative to the nodes in this graph. Final co-ordinates will be determined at sea based on



conditions and traffic.

## Test Cases<sup>1</sup>

Nominal Termination Criteria:

- Correct execution of the plan. Arrived at expected position.
- No aborts.
- An exit sent from AMC to VCS
- VCS jumps into the abort plan.

It should look something like:

```
[VCS][VCS][400][1182204828.787091]Terminating due to watchdog timeout.
```

```
[VCS][VCS][400][1182204828.787140]Sending exit
```

```
[VCS][VCS][400][1182204828.787214]Exiting command dispatcher loop.
```

```
[VCS][VCS][400][1182204828.787313]Terminating Command Dispatcher
```

```
[VCS][VCS][400][1182204828.791884]Terminating State Listener.
```

---

<sup>1</sup> Results noted are based on preliminary analysis.

## **Safety Checks**

### **checkin.0**

This test will execute a canned plan consisting of a call to **getgps** and then a call to **setpoint**. The mission time limit is 100 seconds. This will test that we can start the vehicle and run a canned plan.

Look for:

- Termination in less than 100 seconds
- Nominal Termination

Results: PASSED AS EXPECTED

### **checkin.0.timeout**

The purpose of this test is to verify that the mission timer will force an abort. To do this we employ a version of the base plan with a very short timeout window – 20 seconds. The AMC time limit will remain at 100 seconds.

**Setup: copy init.timer.cfg to init.cfg**

Look for:

- Termination initiated from the VCS after 20 seconds.
- Jumps into the abort plan

**Don't forget: copy init.base.cfg to init.cfg**

Results: PASSED AS EXPECTED

### **checkin.0.terminate**

The purpose of this test is to ensure that if the AMC crashes, the VCS will jump into the abort plan within 10 seconds based on its own watchdog. Revert to standard base plan. Once again use checkin.0. After completion of the gps, kill the AMC.

Look for:

- VCS Jumps into the abort plan.

Results: PASSED AS EXPECTED

### **timing.1**

This test will run a **getgps** and a sequence of **setpoints**, all very short. It is a simple timing test to make sure the commanding and feedback is correct. The vehicle will remain at the surface. The AMC timeout will be 100 seconds.

Results: PASSED AS EXPECTED

### **go.base**

This test will run a **setpoint** and **waypoint** at the surface, followed by a **getgps**.

Results: PASSED AS EXPECTED

## **depth.abort**

This test will verify that the depth envelope behavior will force an abort if the vehicle goes too deep. We will use a canned plan to descend to a target depth of 20 meters but modify the base plan to have a max depth of 10 meters. We can set the AMC timeout to 100 seconds.

**Setup: copy init.depth.cfg to init.cfg**

Look for :

- VCS “DepthEnvelope -- Measured depth 10.064657 over allowable depth 10.000000. Aborting!”
- VCS : Abort request issued by behavior "depthEnvelope"
- AMC: Received a termination event from the VCS

**Don't forget: copy init.base.cfg to init.cfg**

Results: PASSED AS EXPECTED. The AMC received an abort message from the VCS at tick 73.

## ***Simple Mobility***

This group of tests will execute traveling to a destination involving the main behaviors. All tests should terminate with the nominal termination sequence initiated by the AMC watchdog timing out after the last command has finished.

NB: Timing constraints in each .cfg file for a test are based on how long the mission takes given an initial position. This may have to be changed if in fact we are further away from the destination. We tend to keep these bounds tight to limit the possibility of faults leading to a missing vehicle.

### **go.0**

This test is a predefined sequence of commands. No plan generation. Only plan validation on input. The plan will drive briefly at the surface, descend to 5 meters, drive to a waypoint, and then ascend to the surface. It uses **centerNorthing** and **centerEasting**.

- Destination == **center node**.
- Timeout == 600
- Max depth = 5 meters

The commands executed should be (waypoint will change):

```
behavior setpoint {
id=311;duration=6;heading=0.000000;speed=1.500000;verticalMode=pitch;pitch=0.000000; }

behavior descend { id=331;duration=59;horizontalMode=heading;horizontal=0.000000;pitch=-
30.000000;speed=1.500000;maxDepth=5.000000;minAltitude=10.000000; }

behavior waypoint {
id=355;duration=59;northing=4066281.115653;easting=599908.200877;speed=1.500000;depth=5.000000;
}

behavior ascend { id=384;duration=59;pitch=30.000000;speed=1.500000;endDepth=0.000000; }
```

Results: PASSED AS EXPECTED.

### go.1

This test will generate a decomposition of **Action.Go** to descend to a target depth of 5 meters. On termination, the commands to descend will be latched in so the vehicle will continue to descend at a pitch angle of -30 degrees and a speed of 1.5 meters for 10 seconds until the watchdog times out and sends an exit, which will jump into the abort plan. Default headings of 270 (due west) will be used.

- Destination = N/A. Heading due west
- Timeout = 100
- Max Depth = 5

The commands generated should be:

```
behavior setpoint {
id=1139;duration=5;heading=270.000000;speed=1.500000;verticalMode=pitch;pitch=0.000000; }
descend { id=1094;duration=65;horizontalMode=heading;horizontal=270.000000;pitch=-
30.000000;speed=1.500000;maxDepth=5.000000;minAltitude=10.000000; }
```

Look for:

- Max depth in the log to be greater than 5 meters. It should not be more than 10 meters.
- Nominal termination.
- The vehicle should float to the surface.

Results: PASSED AS EXPECTED. Descend completed at tick 61.

### go.2

This test will acquire the **centerNode**. It will use a waypoint mode and a max depth of 5 meters. The heading generated will depend on where the vehicle is.

- Destination = **center node**
- Timeout = 600
- Max Depth = 5

The commands generated should be approximately:

```
behavior setpoint {
id=1140;duration=5;heading=207.731830;speed=1.500000;verticalMode=pitch;pitch=0.000000; }
behavior descend { id=1095;duration=65;horizontalMode=heading;horizontal=207.731830;pitch=-
30.000000;speed=1.500000;maxDepth=5.000000;minAltitude=10.000000; }
behavior waypoint {
id=4807;duration=121;northing=4066281.115653;eastings=599908.200877;speed=1.500000;depth=5.000000; }
```

Look for:

- Arrival at the center node, at least crossing the line perpendicular to the track-line.

- A max depth of approximately 5 meters.

Results: PASSED AS EXPECTED. Waypoint was achieved at tick 120. It was later successfully used on 2 occasions (#28 and #31) to reposition the vehicle taking 160 and 176 ticks respectively.

### go.3

This test will plan for 2 goals. The first will **dive** and the second will **ascend**. It can be run anywhere since no lat/long targets are used. Default headings will drive the vehicle west. The expected depth is 5 meters. It may go slightly deeper transitioning from descend to ascend.

- Destination – N/A. Will head due west.
- Timeout = 100.
- Max Depth = 5.

The commands generated should be:

```
behavior setpoint {
id=1166;duration=5;heading=270.000000;speed=1.500000;verticalMode=pitch;pitch=0.000000; }
behavior descend { id=1121;duration=65;horizontalMode=heading;horizontal=270.000000;pitch=-30.000000;speed=1.500000;maxDepth=5.000000;minAltitude=10.000000; }
behavior ascend { id=5071;duration=17;pitch=30.000000;speed=1.500000;endDepth=0.000000; }
```

Results: PASSED AS EXPECTED. However, the timeout was too short as the ascend command had not quite completed. It was within less than a meter on completion.

### go.5

This test introduces a yoyo. It will make for the **west node** with drive mode set for yoyo and using a minDepth of 10 and a maxDepth of 20. No **checkin** requirements are imposed. This test should be initiated **near the center node**, or at least within a reachable distance from the **west node**.

- Destination = West Node.
- Timeout = 600
- Max Depth = 20

The commands generated should be:

```
behavior setpoint {
id=1145;duration=5;heading=235.449500;speed=1.500000;verticalMode=pitch;pitch=0.000000; }
behavior descend { id=1100;duration=72;horizontalMode=heading;horizontal=235.449500;pitch=-30.000000;speed=1.500000;maxDepth=10.000000;minAltitude=10.000000; }
behavior waypoint_yoyo {
id=5774;duration=155;northing=4066280.065570;eastings=599818.282261;speed=1.500000;minDepth=10.000000;maxDepth=20.000000;minPitch=-15.000000;maxPitch=15.000000;minAltitude=10.000000;maxCycles=100.000000; }
```

Results: PASSED AS EXPECTED. West node achieved at tick 110.

## Checkin Tests

This group of tests verify the checkin action which will get the vehicle to the surface, localize via gps, and then idle at the surface for a designated period of time.

### checkin.1

The vehicle will cycle through a sequence of active checkin windows for the duration of the mission. The separation between checkin's is 1 second. The duration of each is 45 seconds. This test can be run anywhere.

- Destination = N/A. No commanded motion.
- Timeout = 300.
- Max Depth = 0.

The commands generated will be something like:

```
|behavior getgps { id=1821;duration=599;abortOnTimeout=True;minHits=30; }  
|behavior setpoint {  
id=5954;duration=9;heading=0.000000;speed=0.000000;verticalMode=pitch;pitch=10.000000; }  
behavior getgps { id=7640;duration=599;abortOnTimeout=True;minHits=30; }  
behavior setpoint {  
id=11773;duration=9;heading=0.000000;speed=0.000000;verticalMode=pitch;pitch=10.000000; }  
behavior getgps { id=13459;duration=599;abortOnTimeout=True;minHits=30; }  
behavior setpoint {  
id=17592;duration=9;heading=0.000000;speed=0.000000;verticalMode=pitch;pitch=10.000000; }  
behavior getgps { id=19278;duration=599;abortOnTimeout=True;minHits=30; }  
behavior setpoint {  
id=23411;duration=9;heading=0.000000;speed=0.000000;verticalMode=pitch;pitch=10.000000; }  
behavior getgps { id=25043;duration=599;abortOnTimeout=True;minHits=30; }
```

Results: PASSED AS EXPECTED. Note that required idle times ranged from 13 to 17 seconds reflecting the uncertainty in gps duration.

### checkin.2

This test will introduce pre-emption of a transect in order to meet checkin requirements. A MAX\_SEPARATION of 60 seconds and a CHECKIN\_DURATION of 45 seconds will be used. The vehicle will be tasked to make the **center node**. Thus it matters where we run it from to reach it in time.

- Destination = center node
- Timeout = 600
- Max depth = 5

The commands generated should be something like:

```
behavior setpoint {  
id=1249;duration=5;heading=207.731830;speed=1.500000;verticalMode=pitch;pitch=0.000000; }
```

```

behavior descend { id=1204;duration=52;horizontalMode=heading;horizontal=207.731830;pitch=-
30.000000;speed=1.500000;maxDepth=5.000000;minAltitude=10.000000; }

behavior waypoint {
id=4987;duration=28;northing=4066281.115653;easting=599908.200877;speed=1.500000;depth=5.000000
; }

behavior ascend { id=8799;duration=14;pitch=30.000000;speed=1.500000;endDepth=0.000000; }

behavior getgps { id=10577;duration=599;abortOnTimeout=True;minHits=30; }

behavior setpoint {
id=14563;duration=14;heading=0.000000;speed=0.000000;verticalMode=pitch;pitch=10.000000; }

behavior setpoint {
id=16789;duration=5;heading=186.941852;speed=1.500000;verticalMode=pitch;pitch=0.000000; }

behavior descend { id=16744;duration=53;horizontalMode=heading;horizontal=186.941852;pitch=-
30.000000;speed=1.500000;maxDepth=5.000000;minAltitude=10.000000; }

behavior waypoint {
id=20094;duration=32;northing=4066281.115653;easting=599908.200877;speed=1.500000;depth=5.00000
0; }

```

Results: PASSED. Note however that a zero duration waypoint was dispatched for execution. This can occur if the duration in the plan is too short (though not zero). The model should be adjusted to threshold generation of a waypoint with a larger duration. The separation windows in this test are very short allowing for little time to do lateral travel. Because ascend, setpoint and descend use the heading to make for the goal, progress is made all the same.

### **checkin.3**

This will send the vehicle 500 meters north and 500 meters west of the center node. It will employ a **waypoint\_yoyo** drive mode this time, and should have a max separation of 150 seconds with 45 seconds checkin duration. An AMC timeout of 1000 seconds is used. It should switch into yoyo mode at the minDepth of 10 meters.

- Destination = Approximately 700 meters north west of the center node.
- Timeout = 1000
- Max Depth = 20

Results: PASSED. However, we note that when the waypoint terminated the vehicle was much closer to the surface than prescribed by its behavior limits, which called for a minDepth of 10 meters. This suggests a substantial overshoot in the waypoint\_yoyo behavior. This should be followed up with the ops team as it is internal to a behavior.

### ***Path Navigation***

This next set of tests involve commanding the vehicle at a higher level of abstraction – i.e. a path. They also introduce multi-link missions and employ the skipper in planning, thus testing distributed planning among 2 deliberative reactors.

## path.0

Go to the **center node** and **check in when done**. Will travel using a waypoint and a depth of 10 meters. The checkin window will be 50 seconds. No separation limits are imposed.

- Destination = center node
- Timeout = 1000
- Max Depth = 10

Results: PASSED. The target location was obtained at tick 499. A checkin was then executed which ascended, did a getgps, and setpoint to meet the required checkin duration.

## path.1

This test interleaves path level planning and checkin window management to pre-empt go actions. The mission is directed to the **north node** with a max separation of 100 seconds and a check-in duration of 100 seconds.

- Destination = north node
- Timeout = 600
- Max Depth = 10

Result: PASSED. Ended on a checkin at tick 835. 3 waypoint commands were executed to reach the goal, interleaved with checkins.

## path.2

This test will head to the **south node** with a max separation of 100 seconds and a checkin duration of 40 seconds. AMC timeout is 1000 seconds. It utilizes the **mission manager**, so it is primarily a test of distributed planning/synchronization.

- Destination = south node.
- Timeout = 1000
- Max Depth = 10

Results: PASSED. This test was tried twice (#21 and #23) from different initial positions and it passed on both occasions. On the first run, the goal was reached at tick 736 after 3 checkins and 4 waypoints. On the second run, the goal was accomplished at tick 988 after 5 waypoints and 5 checkins.

## path.3

This test introduces a transition from one target node to another. The max separation is 240 seconds and the minimum checkin duration is 45 seconds.

- Destinations – first to the west node, then to the north node
- Timeout = 1800
- Max Depth = 10

Results: PASSED. This test was also tried twice (#22 and #32) and passed on both occasions. On the first run, the west node was reached at tick 466. The next waypoint command was dispatched at tick 471 with no commands in between. This illustrated that no checkin was required at that time and there was no depth change required so the vehicle did not have to ascend or descend. The north node was achieved at tick 799. In the second run, the vehicle was closer to the west node. Thus the first goal was accomplished earlier (tick 120). Once again the transition on the path happened underwater without a checkin or commanded depth change. The mission was completed at tick 442.

## ***Mission Manager***

These tests introduce mission commanding at the level of science goals and operator. The science goals stipulate a sampling mode to take and spatial and temporal constraints on where to take it. The operator goal is to be at a designated pickup point at the end of the mission. All goals have a priority. Priority for the operator is the highest. Unless over-subscribed, the priority has no effect. The mission manager will handle all the path planning and the navigator will map these to actions and commands for execution based on sensor data. These tests integrate a lot of deliberative and reactive planning, and a great deal of co-ordination between reactors.

### **mission.0**

Assumes we are at the **center node**. The goals are:

- Science Goal 0: waypoint\_yoyo between 10 and 20 meters, from **center node to north node**.
- Science Goal 1: waypoint\_yoyo between 15 and 25 meters, from **east node to south node**.
- Operator Goal: Pickup at **west node**.

Constraints:

- Mission Time  $\leq$  1800
- MAX\_SEPARATION = 240
- CHECKIN\_DURATION = 40

Results (FAILED):

- The first run (#23), we observed that the vehicle commenced the mission as expected but completed after only 89 seconds. A quick look at the TREX log output indicated the waypoint\_yoyo completed indicating it reached the north node. We pulled the logs for detailed analysis and re-ran the test again (#24). The second run also terminated prematurely in a similar way. A quick analysis suggested that the mission was being started too close to the north node and thus the duration lower bound derived from the path network was too high, causing a plan failure. This is a bug in the model. We should not be assuming we are at the center node but rather impose a functional relationship between the path and the position timelines.

- We repositioned the vehicle by executing path.2. We re-ran the test again this time with a shorter MAX\_SEPARATION (120) and a longer CHECKIN\_DURATION (100). This was done to make it easier to track the vehicle as the tracking from the ship was unreliable at such shallow depths. The mission ran for a total of 730 seconds. It terminated with a plan failure when checking in having successfully accomplished the first science goal.

#### Analysis:

- Path information is not being derived correctly from position data. Currently the agent can assert that it is at a given node and not be contradicted by the position data. It should be straightforward to address this in the model.
- The duration bounds for a yoyo are too short. The model currently computes a distance estimate regardless of the drive mode. Obviously this will give too tight a bound if we do not account for the pitch angle when yo-yoing as opposed to level flight. This is not the cause of the early failure but it is something that can and should be corrected in the model.
- The end of the mission was co-incident with the transition from completion of the first science goal and on to the next transact in the path. Curiously, the log showed that the agent used a setpoint and a descend at the start of the next leg. This was surprising since they are only injected if the vehicle is at the surface. Analysis of the logs generated in playback show that the navigator did indeed believe it was at the surface even though it's depth was about 2.3 meters. This is not line with our revised policy. It turns out the decomposition logic in the model does not reflect the revised surface semantics. This is a model bug that can be trivially addressed.
- Having initiated a descend, the mission then switched to a getgps. This was a surprise since having descend for a time it seemed unlikely the vehicle was at the surface. Thus it should have ascended first. A close examination of the logs after replaying shows getgps is generated as part of a checkin, and planned out before the checkin starts. The ascend is inserted based on position data during synchronization. We should defer refinement of the checkin action until it starts as is the case with the go action. This is a simple change in the model.
- The root cause of the failure was due to a squeezing of the duration bounds of a descend by a pending checkin. As a result, when the descend terminated, it had not reached its expected depth as required by the model. Thus the plan failed. This is an issue of *controllability*. The demands of the plan squeezed the time window for descent which included a flexible duration to reflect the inherent uncertainty of the model in this regard. There are a number of ways we can approach this kind of thing:
  - Just re-plan when it fails
  - Get smarter about skipping a dive when we are so close to a check-in window
  - Explicitly prohibit the shrinking of uncontrollable bounds thus recognizing the potential fault earlier

- Make the model more accurate in computing tighter bounds
- Combinations of the above

### **mission.1**

Assumes we are at the west node. The goals are:

- Science Goal 0: waypoint\_yoyo between 10 and 20 meters, from **far west node to south node**.
- Science Goal 1: waypoint\_yoyo between 15 and 25 meters, from **east node to center node**.
- Science Goal 2: waypoint\_yoyo between 10 and 25 meters, from **west node to north node**.
- Operator Goal: Pickup at **center node**.

Constraints:

- Mission Time  $\leq 3600$
- MAX\_SEPARATION = 240
- CHECKIN\_DURATION = 40

Results: This test was not attempted as it had not completed successfully in simulation and since mission.0 had failed.

## **Appendix A: System Operation**

### **Machine Setup**

- MVC IP\_ADDRESS – 134.89.32.39
- MVC Host Name – mvc-dmo1.shore.mbari.org
- State Publisher Socket Port: 8004
- VcsServer Socket Port: 8002
- Trex2 IP\_ADDRESS – 134.89.32.39

### ***MVC Environment Variables***

- AUV
- AUV\_CONFIG\_DIR
- AUV\_LOG\_DIR
- AUV\_PLAN\_DIR
- See mvc-dmo1:~/auv\_autonomy/configSrc

### ***TREX Environment Variables***

- See TREN/amcConfig, TREN/devConfig
- ROOT\_DIR. Make sure this is set to the parent directory containing
- PLASMA\_HOME
- TREN\_HOME
- AMC\_HOME
- AUV\_HOME
- PATH
- LD\_LIBRARY\_PATH

### ***Configuration Files & Scripts***

- vcsServer.cfg
- statePublisher.cfg
- abortPlan.cfg
- devices.cfg
- vcs.cfg

### ***Dovekie Network Setup***

To have dovekie (autonomy laptop) talk directly to mvc-dmo1 we need to do the following.

1. on trex2:/etc/rc.local add (this has been done already)

```
route add -network 10.250.32.0 gw 134.89.32.26 netmask 255.255.255.0 dev eth0
```

(first IP is dmo1's assigned 10.X address, second IP is dmo-1's assigned cabled IP). This ensures a static route to the 10.X between trex2 and dmo1

2. on dovekie: /etc/rc.local add

```
ifconfig eth0:0 10.250.32.4 netmask 255.255.255.0
```

(first IP is dovekie's assigned IP on the 10.X Freewave modem network. **Note 0 is zero**)

3. check dovekie's address with:

```
% ifconfig eth0:0
```

(you should see the 10.X address on this interface)

4. when the cable connection with dmo1 is severed and the Freewave modem needs to be activated then on dovekie do:

```
% route add 134.89.32.39 gw 10.250.32.1
```

which physically adds a route to the 10.X gateway

5. check to see if this is actually done by:

```
% route -v
```

which should show something like the following in a tabular output

```
trex2  10.250.32.1  UGH .....
```

Connection between dovekie and trex2 via dmo1 should be working at this stage.

Dovekie is set with a static IP for ship-based operation. To change to shore-based networking, contact Kanna if the following does not work for you.

#### Addendum:

To change between ship-based (static) routing and shore based (DHCP or dynamic) routing the two files you will need to make changes to are

1. /etc/resolve.conf – ensure that shore based it has at least two nameserver addresses while ship based it has only “search localdomain”
2. /etc/sysconfig/network-scripts/ifcfg-eth0 – ensure shore based has the BOOTPROTO set to ‘dhcp’ while ship based it is set to ‘static’

To stop/start network do the following; you will need this only when you want to make changes to the network addresses and/or move the laptop from one subnet to another:

```
% /etc/init.d/network stop  
% /etc/init.d/network start
```

## Miscellaneous

- Bootup:
  - The magnet should be in place.
  - Power on the power supply by pressing the red button (in the lab). Expect voltage of  $\leq 35$  V and current approx 2 Amps. There is an led indicator if remote mode is on. This can be an indicator of someone else using it so check with Doug, Duane or Hans.
  - If LED below current indicator is on, power down & call Doug.
  - Use a ping for mvc-dmo1 to wait for it to come up.
  - Telnet mvc-dmo1 (dorado1/dorado1)
  - `qtalk -m /dev/ser13`
  - `!a <ENTER>` // To turn on trex2
  - `!f <ENTER>` // To turn off trex2
  - `^a` to exit qtalk.
  - Ping trex2 to wait to come up. It can take a while. 134.89.32.39
  - telnet trex2 (root/dorado1)
  - remove the magnet before running a mission
- Shutdown
  - Insert magnet
  - Press red button on power supply. Make sure u press in all the way.
- Clearing shared memory
  - If the vcsServer terminates with resource temporarily unavailable messages, you need to nuke files for shared memory access
  - `/dev/shmem` may contain files of the for “\***Shmem**\*”
  - `rm /dev/shmem/*Shmem*`
- Starting vcsServer
  - `cd $AUV/bin`
  - `./vcsServer -v -sim` // for simulation
  - `./vcsServer -v` // for real missions
- Killing vcsServer
  - `slay supervisor`
  - `^c`
- Deployment On dovekey :
  - `cd amc/TREX`

- source devConfig
- jam -sVARIANTS=OPTIMIZED amc
- amcDeploy
- scp amc.tar.gz trex2:
- Deployment On trex2:
  - cd amc/TREX
  - source amcConfig
  - cd \$AMC\_HOME/deploy
  - gunzip -c ~/amc.tar.gz | tar xvf -

