# DARTSim TCP-based Interface

The simulation provides a monitoring and execution interface through a TCP port. Commands are sent to TCP port 5418. A command consists of a command name followed by space-separated parameters, if the commands has parameters. If the command is not recognized, DARTSim replies with a text line with the prefix "error:", followed by an error message. Otherwise, the reply is a string encoding the return value of the command.

The details of the commands in DARTSim's API follow.

## step

Executes one simulation step.

### Parameters

- *tacticsList*: List of tactic names to execute encoded as a JSON list of strings. If no tactics must be executed, the list should be empty (i.e., [ ]).
- *decisionTime*: Time that the adaptation manager took to make an adaptation decision as a double.

### Return value

String: ("true" if a target was detected with the downward-looking sensor, otherwise "false")

### Example

Command: `step ["DecAlt","GoLoose"] 132.0`
Reply: `false`

## getState

Get the state of the team.

### Parameters

None

### Return value

JSON structure with the following fields:

- *unsigned altitudeLevel*: Altitude level of the team.
- *int directionX*: Horizontal direction of the team. This is also the horizontal direction of the forward-looking sensors. -1, 0 or +1 to indicate the horizontal direction of travel.
- *int directionY*: Vertical direction of the team. This is also the vertical direction of the forward-looking sensors. -1, 0 or +1 to indicate the vertical direction of travel.
- *bool ecm*: Whether electronic countermeasures (ECM) are on.
- *unsigned formation*: Current formation: 0 for loose formation, 1 for tight formation.

- *unsigned ttcDecAlt*: Time in periods to complete altitude decrease, 0 means not executing.
- *unsigned ttcDecAlt2*: Time in periods to complete altitude decrease 2, 0 means not executing.
- *unsigned ttcIncAlt*: Time in periods to complete altitude increase, 0 means not executing.
- *unsigned ttcIncAlt2*: Time in periods to complete altitude increase 2, 0 means not executing.

### Example

Command: `getState`

Reply: `{"altitudeLevel": 1, "directionX": 1, "directionY": 0, "ecm": false, "formation": 0, "positionX": 17, "positionY": 0, "ttcDecAlt": 0, "ttcDecAlt2": 0, "ttcIncAlt": 0, "ttcIncAlt2": 0}`

## getParameters

Get simulation parameters

### Parameters

None

### Return value

JSON structure with the following fields:

- *unsigned altitudeLevels:* The number of altitude levels.
- *unsigned changeAltitudeLatencyPeriods:* Latency of the tactic to change altitude.
- *double destructionFormationFactor:* Factor that specifies how much the destruction probability is reduced when the team flies in tight formation.
- *unsigned mapSize*: This is the size of the map in route segments.
- *bool optimalityTest:* When this is true, the sensors behave deterministically, in the sense that if a target is in range it is detected. The same with threats.
- *bool squareMap*: Whether the map is square.
- *double targetDetectionFormationFactor:* Factor that specifies how much the detection probability is reduced when the team flies in tight formation.
- *unsigned targetSensorRange:* The detection range for the sensor specified in altitude levels.
- *double targetSensorFNR:* False negative ratio of the target sensor.
- *double targetSensorFPR:* False positive ratio of the target sensor.
- *unsigned threatRange:* The range for a threat specified in altitude levels.
- *double threatSensorFNR:* False negative ratio of the threat sensor.
- *double threatSensorFPR:* False positive ratio of the threat sensor.

### Example

Command: `getParameters`

Reply: `{"altitudeLevels": 4, "changeAltitudeLatencyPeriods": 1, "destructionFormationFactor": 1.5, "mapSize": 40,`

```
"optimalityTest": false, "squareMap": false,
"targetDetectionFormationFactor": 1.2, "targetSensorFNR":
0.14999999999999999, "targetSensorFPR":
0.10000000000000001, "targetSensorRange": 4, "threatRange":
3, "threatSensorFNR": 0.14999999999999999,
"threatSensorFPR": 0.10000000000000001}
```


### finished

Checks whether the simulation has finished

#### Parameters

None

#### Return value

String: ("true" if the simulation has finished, otherwise "false")

#### Example

Command: `finished`
Reply: `false`

### readForwardThreatSensor

Reads the forward-looking threat sensor and reports whether a threat was sensed in each cell in the straight forward path

#### Parameters

*unsigned cells*: Number of cells to sense.

#### Return value

JSON structure representing a list of Booleans

#### Example

Command: `readForwardThreatSensor(5)`
Reply: `[true, false, false, true, false]`

### readForwardTargetSensor

Reads the forward-looking target sensor and reports whether a target was sensed in each cell in the straight forward path

#### Parameters

*unsigned cells*: Number of cells to sense.

#### Return value

JSON structure representing a list of Booleans

#### Example

Command: `readForwardTargetSensor(5)`
Reply: `[false, true, true, false, false]`

### readForwardThreatSensorForObservations

Reads several observations with the forward-looking threat sensor. This method is for convenience, and required to implement adaptation managers compatible with the original version of DARTSim. It gets multiple observations of each cell instead of just one

#### Parameters

- *unsigned cells*: Number of cells to sense.
- *unsigned numOfObservations*: Number of observations to take for each cell.

#### Return value

JSON structure representing a list of lists of Booleans

#### Example

Command: `readForwardThreatSensorForObservations(4, 2)`

Reply: `[[false, true], [false, false], [false, false], [false, false]]`

### readForwardTargetSensorForObservations

Reads several observations with the forward-looking target sensor. This method is for convenience, and required to implement adaptation managers compatible with the original version of DARTSim. It gets multiple observations of each cell instead of just one

#### Parameters

- *unsigned cells*: Number of cells to sense.
- *unsigned numOfObservations*: Number of observations to take for each cell.

#### Return value

JSON structure representing a list of lists of Booleans

#### Example

Command: `readForwardTargetSensorForObservations(5, 3)`

Reply: `[[false, false, false], [false, false, false], [false, false, true], [false, false, false], [false, false, false]]`

### getScreenOutput

Gets visual trace, which depicts a 2D side view of the team's route

#### Parameters

None

#### Return value

String

#### Example

Command: `getScreenOutput`

Reply: `"# ## #  # # ##### # #  ## ##### #\n #  # # # #
#  # # #  # ## #  # \n  #  ## ##
\n  \n ^  ^^
^ ^^  \n  X  TT
T  \n"`

| Symbol | Meaning |
|---|---|
| # | loose formation |
| * | tight formation |
| @ | loose formation, ECM on |
| 0 | tight formation, ECM on |
| ^ | threat |
| T | target (not detected) |
| X | target (detected) |

### getResults

Get result parameters

#### Parameters

None

#### Return value

JSON structure with the following fields:

- *double decisionTimeAvg*: Average decision time (if reported).
- *double decisionTimeVar*: Variance of decision time (if reported).
- *bool destroyed*: Whether team was destroyed.
- *int positionX*: If the team was destroyed, x-coordinate for the position where it happened.
- *int positionY*: If the team was destroyed, y-coordinate for the position where it happened.
- *bool missionSuccess*: Whether the mission was completed successfully.
- *unsigned targetsDetected*: Number of targets detected.

#### Example

Command: `getResults`

Reply: `{"decisionTimeAvg": 221.75, "decisionTimeVar": 49176.800000000003, "destroyed": false, "destruction positionX": 39, "destruction positionY": 0, "missionSuccess": true, "targetsDetected": 2}`