Python, MATLAB, Scilab & GNU Octave API for NATS

Updated: 09.10, 2019

NATS Client API

No.	Type	Method and Description
1	`EntityInterface`	**`getEntityInterface()`** Returns a reference to the EntityInterface.
2	`EnvironmentInterface`	**`getEnvironmentInterface()`** Returns a reference to the EnvironmentInterface.
3	`EquipmentInterface`	**`getEquipmentInterface()`** Returns a reference to the EquipmentInterface.
4	`SafetyMetricsInterface`	**`getSafetyMetricsInterface()`** Returns a reference to the SafetyMetricsInterface.
5	`SafetyMetricsInterface`	**`getSafetyMInterface()`** Returns a reference to the SafetyMetricsInterface, an alias for Scilab platform (Due to syntax restrictions).
6	`SimulationInterface`	**`getSimulationInterface()`** Returns a reference to the SimulationInterface.
7	`void`	**`disConnect()`** Close the connection with the NATS Server.
8	`void`	**`login(String authenticationCode)`** User log in with authentication code (to be used for multi-user simulation mode).

SimulationInterface API

No.	Type	Method and Description
1	`void`	**`clear_trajectory()`** Cleanup the trajectory data.
2	`float`	**`get_curr_sim_time()`** Get the current simulation timestamp.
3	`long`	**`get_sim_id()`** Get the simulation id.

4	`int`	**`get_runtime_sim_status()`**
		Get the runtime status of the trajectory propagation.
		Value definition:
		NATS_SIMULATION_STATUS_READY = 0
		NATS_SIMULATION_STATUS_START = 1
		NATS_SIMULATION_STATUS_PAUSE = 2
		NATS_SIMULATION_STATUS_RESUME = 3
		NATS_SIMULATION_STATUS_STOP = 4
		NATS_SIMULATION_STATUS_ENDED = 5
		When the trajectory propagation finishes, the status will be changed to NATS_SIMULATION_STATUS_ENDED.

5	`void`	**`pause()`** Pause the trajectory propagation process. This function is disabled in real-time simulation mode.

6	`void`	**`resume()`** Resume the trajectory propagation process.
7	`void`	**`resume(long t_duration)`** Resume the trajectory propagation process and process data for a specified duration of time (in seconds).
8	`int`	**`setupSimulation(long t_total_propagation_period, long t_step)`** Setup the trajectory propagation process. Description of the arguments: t_total_propagation_period: Total period of time of propagation in seconds. t_step: Time step in seconds. For surface ground traffic, the recommended propagation time step is 1 second.
9	`void`	**`start()`** Start the trajectory propagation process.
10	`void`	**`start(long t_duration)`** Start the trajectory propagation process for specified duration, in seconds.
11	`void`	**`startRealTime()`** Start the real-time trajectory propagation. NATS Server runs trajectory propagation with 30-second time step, synchronized with real-time clock.
12	`void`	**`startRealTime_singleUser()`** Start the real-time trajectory propagation while in single-user mode. NATS Server runs trajectory propagation with 30-second time step, synchronized with real-time clock. Aircraft state data can be imported from an external aircraft simulator to the NATS Server. Please refer to the *XPlane* simulation example for the details.
13	`void`	**`stop()`** Stop the trajectory propagation process.
14	`void`	**`write_trajectories(String output_file)`** Write trajectory data into a file. File format supported: .csv, .kml, .xml
15	`void`	**`request_aircraft(String ac_id)`** Request aircrafts from NATS Server which is the administrator for multi-user simulation. The aircraft pertaining to the callsign given in the argument ac_id will be assigned to the client based on First-Come-First-Serve policy.
16	`void`	**`request_groundVehicle(String gv_id)`** Request ground vehicles from NATS Server which is the administrator for multi-user simulation. The ground vehicle pertaining given in the argument gv_id will be assigned to the client based on First-Come-First-Serve policy.

17	void	**externalAircraft_create_trajectory_profile(String ac_id, String ac_type, String origin_airport, String destination_airport, float cruise_altitude_ft, float cruise_tas_knots, double latitude_deg, double longitude_deg, double altitude_ft, double rocd_fps, double tas_knots, double course_deg, String flight_phase)** Create the trajectory profile and set the initial state of an external aircraft in NATS.
18	void	**externalAircraft_inject_trajectory_state_data(String ac_id, double latitude_deg, double longitude_deg, double altitude_ft, double rocd_fps, double tas_knots, double course_deg, String flight_phase, long timestamp_utc_millisec)** Send external aircraft state data from the client to the server.
19	void	**requestDownloadTrajectoryFile()** Request the download of the latest trajectory file from the NATS Server. Due to the potential for simultaneous file-downloading requests from users, the file downloading process may not start immediately.

Simulation Status Enum Values

Values
NATS_SIMULATION_STATUS_READY **NATS_SIMULATION_STATUS_START** **NATS_SIMULATION_STATUS_PAUSE** **NATS_SIMULATION_STATUS_RESUME** **NATS_SIMULATION_STATUS_STOP** **NATS_SIMULATION_STATUS_ENDED**

EquipmentInterface API

No.	Type	Method and Description
1	`AircraftInterface`	**`getAircraftInterface()`** Returns a reference to the AircraftInterface.
2	`GroundVehicleInterface`	**`getGroundVehicleInterface()`** Returns a reference to the GroundVehicleInterface.
3	`CNSInterface`	**`getCNSInterface()`** Returns a reference to the CNSInterface.
4	`BADADataInterface`	**`getBADADataInterface()`** Returns a reference to the BADADataInterface.

AircraftInterface API

No.	Type	Method and Description
1	`int`	**`load_aircraft(String trx_file, String mfl_file)`** Load aircraft data.
2	`boolean`	**`validate_flight_plan_record(String string_track, String string_fp_route, int mfl_ft)`** Validator of flight plan record.
3	`int`	**`release_aircraft()`** Cleanup aircraft data.
4	`String[]`	**`getAircraftIds(float minLatitude, float maxLatitude, float minLongitude, float maxLongitude, float minAltitude_ft, float maxAltitude_ft)`** Get IDs of all aircraft within the min/max range of latitude, longitude and/or altitude ranges.
5	`String[]`	**`getAllAircraftId()`** Get the complete list of all aircraft IDs in the NATS simulation.
6	`Aircraft`	**`select_aircraft(String aircraft_id)`** Get an aircraft object with aircraft ID.
7	`int`	**`synchronize_aircraft_to_server(Aircraft aircraft)`** Push aircraft object to the server and synchronize the data. Return value indicates the server operation response: 0 is success. 1 indicates error.

Aircraft Instance API

No.	Type	Method and Description
1	`int`	**`delay_departure(int seconds)`** Postpone the departure time of the current aircraft by certain seconds. If the aircraft has already departed, the departure time will not be changed.
2	`String`	**`getAcid()`** Get aircraft ID. Example: UA555
3	`float`	**`getAltitude_ft()`** Get the current altitude in feet.
4	`float`	**`getCruise_alt_ft()`** Get the cruise altitude in feet.
5	`float`	**`getCruise_tas_knots()`** Get cruise speed.
6	`float`	**`getDeparture_time_sec()`** Get departure time in seconds.

7	`float`	**`getDestination_airport_elevation_ft()`** Get the elevation of the destination airport.
8	`int`	**`getFlight_phase()`** Get current flight phase. Flight phase is presented as an integer in the range 1-25. Please refer to “Flight Phase Enum Values” for the definition of each phase.
9	`float[]`	**`getFlight_plan_latitude_array()`** Get the latitude array of the flight plan.
10	`int`	**`getFlight_plan_length()`** Get the number of records in the flight plan.
11	`float[]`	**`getFlight_plan_longitude_array()`** Get the longitude array of the flight plan.
12	`String[]`	**`getFlight_plan_waypoint_name_array()`** Get the array of waypoint names in the flight plan.
13	`String[]`	**`getFlight_plan_alt_desc_array()`** Get the array of flight plan altitude constraint description. Refer to ARINC 424-18 Section 5.29 for details.
14	`double[]`	**`getFlight_plan_alt_1_array()`** Get the array of flight plan altitude first bound. Refer to ARINC 424-18 Section 5.30 for details.
15	`double[]`	**`getFlight_plan_alt_2_array()`** Get the array of flight plan altitude second bound. Refer to ARINC 424-18 Section 5.30 for details.
16	`double[]`	**`getFlight_plan_speed_limit_array()`** Get the array of flight plan speed limits. Refer to ARINC 424-18 Section 5.72 for details.
17	`String[]`	**`getFlight_plan_speed_limit_desc_array()`** Get the array of flight plan speed limit constraint description. Refer to ARINC 424-18 Section 5.261 for details.
18	`float`	**`getFpa_rad()`** Get the current flight path angle, radians.
19	`float`	**`getCourse_rad()`** Get the current course, radians.
20	`int`	**`getLanded_flag()`** Get the flag value indicating if the aircraft has landed.
21	`float`	**`getLatitude_deg()`** Get the current latitude, degrees.
22	`float`	**`getLongitude_deg()`** Get the current longitude, degrees.
23	`float`	**`getOrigin_airport_elevation_ft()`** Get the elevation of the origin airport, feet.
24	`float`	**`getRocd_fps()`** Get the rate of climb or descent in feet per second.
25	`int`	**`getSector_index()`** Get the current sector index.
26	`int`	**`getTarget_waypoint_index()`** Get the array index of the target waypoint in the flight plan

27	String	**getTarget_waypoint_name()** Get the target waypoint name.
28	float	**getTas_knots()** Get the current speed.
29	int	**getToc_index()** Get the flight plan array index of the top-of-climb waypoint.
30	int	**getTod_index()** Get the flight plan array index of the top-of-descent waypoint.
31	void	**setAltitude_ft(float altitude_ft)** Set a new value of altitude in feet.
32	void	**setCruise_alt_ft(float cruise_alt_ft)** Set a new value of cruise altitude in feet.
33	void	**setCruise_tas_knots(float cruise_tas_knots)** Set a new value of cruise speed.
34	void	**setFlight_plan_latitude_deg(int index, float latitude_deg)** Set the latitude of the n-th waypoint.
35	void	**setFlight_plan_longitude_deg(int index, float longitude_deg)** Set the longitude of the n-th waypoint.
36	void	**setCourse_rad(float course_rad)** Set a new value of course.
37	void	**setLatitude_deg(float latitude_deg)** Set a new value of latitude.
38	void	**setLongitude_deg(float longitude_deg)** Set a new value of longitude.
39	void	**setRocd_fps(float rocd_fps)** Set a new value of rate of climb or descent in feet per second.
40	void	**setTarget_waypoint_latitude_deg(float latitude_deg)** Set a new value for the target (Next) waypoint latitude..
41	void	**setTarget_waypoint_longitude_deg(float longitude_deg)** Set a new value for the target (next) waypoint longitude.
42	void	**setTas_knots(float tas_knots)** Set a new value for speed, in knots.

GroundVehicle Interface API

No.	Type	Method and Description
1	int	**load_groundVehicle(String trx_file)** Load all the ground vehicles from the TRX file.
2	int	**release_groundVehicle()** Clear all ground vehicle drive plan data.
3	String[]	**getAllGroundVehicleIds()** Get callsigns of all ground vehicles loaded in NATS.
4	GroundVehicle	**select_groundVehicle(String groundVehicleId)**

		Get GroundVehicle object for a given vehicle callsign.
5	`String[]`	**`GetAssignedGroundVehicleIds()`** Get IDs of ground vehicles which are assigned to current session user.
6	`String[]`	**`getAssignedGroundVehicleIds(String username)`** Get IDs of ground vehicles which are assigned to the user.
7	`int`	**`externalGroundVehicle_create_trajectory_profile(String groundVehicleId, String aircraft, String airport, float latitude, float longitude, float speed, float course)`** Create profile of an external ground vehicle.
8	`int`	**`externalGroundVehicle_inject_trajectory_state_data(String groundVehicleId, String aircraftInService, float latitude, float longitude, float speed, float course)`** Update profile of an existing external ground vehicle.

GroundVehicle Instance API

No.	Type	Method and Description
1	`String`	**`getGvid()`** Get ground vehicle ID.
2	`String`	**`getAirportId()`** Get airport ICAO code of the ground vehicle.
3	`String`	**`getAircraftInService()`** Get aircraft ID being serviced by ground vehicle.
4	`boolean`	**`getFlag_external_groundvehicle()`** Get the flag to determine if the ground vehicle is external. TRUE if the ground vehicle is external.
5	`String`	**`getAssigned_user()`** Get the assigned user
6	`float`	**`getLatitude()`** Get the current latitude, degrees.
7	`void`	**`setLatitude(float latitude)`** Set the new value to current latitude, degrees.
8	`float`	**`getLongitude()`** Get the current longitude, degrees.
9	`void`	**`setLongitude(float longitude)`** Set the new value to current longitude, degrees.
10	`float`	**`getAltitude()`** Get the current altitude in feet.
11	`float`	**`getSpeed()`** Get the current speed.
12	`void`	**`setSpeed(float speed)`** Set the current speed
13	`float`	**`getCourse()`** Get the current course.
14	`void`	**`setCourse(float course)`** Set the new value to the current course.

15	`float`	**`getDeparture_time()`** Get the departure time.
16	`float[]`	**`getDrive_plan_latitude_array()`** Get the array of latitude of the drive plan.
17	`float[]`	**`getDrive_plan_longitude_array()`** Get the array of longitude of the drive plan.
18	`int`	**`getDrive_plan_length()`** Get the number of records in the drive plan.
19	`String[]`	**`getDrive_plan_waypoint_name_array()`** Get the array of waypoint names of the drive plan.
20	`int`	**`getTarget_waypoint_index()`** Get the array index of the drive plan data corresponding to the target waypoint.
21	`String`	**`getTarget_waypoint_name()`** Get the name of the drive plan data corresponding to the target waypoint.
22	`void`	**`setDrive_plan_latitude(int index, float latitude)`** Set the latitude of the n-th drive plan waypoint, degrees.
23	`void`	**`setDrive_plan_longitude(int index, float longitude)`** Set the longitude of the n-th drive plan waypoint, degrees.

CNSInterface API

No.	Type	Method and Description
1	`double[]`	**`getLineOfSight(double observerLat, double observerLon, double observerAlt, double targetLat, double targetLon, double targetAlt)`** Computes the line of sight between source and target, returns range, azimuth, and elevation along with masking due to terrain or earth's curvature. observerLat: Latitude at the observer's location, degrees. observerLon: Longitude of observer's location, degrees. observerAlt: Observer's altitude, feet. targetLat: Latitude at the target's location, feet. targetLon: Longitude of target's position, feet. targetAlt: Altitude of target, feet. Array as (Range (ft), Azimuth (degree), Elevation(degree), Masking (boolean)) of target relative to the observer. The Masking boolean can assume values: 0: No Masking, 1: Terrain Masking, 2: Masking due to the curvature of Earth.
2	`int`	**`setNavigationLocationError(String aircraftId, String parameter, double bias, double drift, double scaleFactor, double noiseVariance, int scope)`** Sets Latitude/Longitude navigation errors for aircraft Navigation System. parameter: String containing "LATITUDE" or "LONGITUDE". bias: Bias to be applied to original value. drift: Drift to be applied to original value multiplied by flight time. scaleFactor: scale factor error that would lead to erroneous instrument values. noiseVariance: Variance of noise to be applied, assuming zero mean Gaussian distribution. scope: 0 for errors to reflect on flight deck systems only, 1 to include errors in

		the ADS-B transmission of the aircraft states.
3	int	**`setNavigationAltitudeError(String aircraftId, double bias, double noiseVariance, int scope)`** Sets altitude errors in the aircraft Navigation System. bias: Bias to be applied to original value. noiseVariance: Variance of noise to be applied, assuming zero mean Gaussian distribution. scope: 0 for errors to reflect on flight deck systems only, 1 to include errors in the ADS-B transmission of the aircraft altitude.
4	int	**`setRadarError(String airportId, String parameter, double originalValue, double bias, double noiseVariance, int scope)`** Applies range, elevation, azimuth errors to the ground radar at an airport. airportId: ICAO code of airport parameter: String containing RANGE, ELEVATION, or AZIMUTH originalValue: The initial true value of the parameter bias: Bias to be applied to original value. noiseVariance: Variance of noise to be applied, assuming zero mean Gaussian distribution. scope: 0 for errors in the ground systems only, 1 to include transmission to aircraft.

BADADataInterface API

No.	Type	Method and Description
1	double	**`getBADA_cruiseTas(String ac_type, double altitude_ft)`** Get cruise speed.
2	double	**`getBADA_climbRate_fpm(String ac_type, double flight_level, String bada_mass)`** Get climb rate in feet per minute.
3	double	**`getBADA_climbTas(String ac_type, double altitude_ft)`** Get climb speed.
4	double	**`getBADA_descentRate_fpm(String ac_type, double flight_level, String bada_mass)`** Get descent rate in feet per minute.
5	double	**`getBADA_descentTas(String ac_type, double altitude_ft)`** Get descent speed.

Flight Phase Enum Values

Values
`FLIGHT_PHASE_ORIGIN_GATE` **`FLIGHT_PHASE_PUSHBACK`** **`FLIGHT_PHASE_RAMP_DEPARTING`**

```
FLIGHT_PHASE_TAXI_DEPARTING

FLIGHT_PHASE_RUNWAY_THRESHOLD_DEPARTING

FLIGHT_PHASE_TAKEOFF

FLIGHT_PHASE_CLIMBOUT

FLIGHT_PHASE_HOLD_IN_DEPARTURE_PATTERN

FLIGHT_PHASE_CLIMB_TO_CRUISE_ALTITUDE

FLIGHT_PHASE_TOP_OF_CLIMB

FLIGHT_PHASE_CRUISE

FLIGHT_PHASE_HOLD_IN_ENROUTE_PATTERN

FLIGHT_PHASE_TOP_OF_DESCENT

FLIGHT_PHASE_INITIAL_DESCENT

FLIGHT_PHASE_HOLD_IN_ARRIVAL_PATTERN

FLIGHT_PHASE_APPROACH

FLIGHT_PHASE_FINAL_APPROACH

FLIGHT_PHASE_GO_AROUND

FLIGHT_PHASE_TOUCHDOWN

FLIGHT_PHASE_LAND

FLIGHT_PHASE_EXIT_RUNWAY

FLIGHT_PHASE_TAXI_ARRIVING

FLIGHT_PHASE_RUNWAY_CROSSING

FLIGHT_PHASE_RAMP_ARRIVING

FLIGHT_PHASE_DESTINATION_GATE

FLIGHT_PHASE_LANDED
```

EnvironmentInterface API

No.	Type	Method and Description
1	`void`	**`load_rap(String wind_dir)`** Load wind RAP file. RAP: NOAA Rapid Refresh wind data
2	`int`	**`release_rap()`** Clean up the RAP data.
3	`AirportInterface`	**`getAirportInterface()`** Returns a reference to the AirportInterface.
4	`TerrainInterface`	**`getTerrainInterface()`** Returns a reference to the TerrainInterface.
5	`TerminalAreaInterface`	**`getTerminalAreaInterface()`** Returns a reference to the TerminalAreaInterface.
6	`WeatherInterface`	**`getWeatherInterface()`** Returns a reference to the WeatherInterface.
7	`String[]`	**`getCenterCodes()`** Returns a String array of all center codes.
8	`String`	**`getCurrentCenter(String aircraftId)`** Returns the center where the given aircraft is located.
9	`String[]`	**`getFixesInCenter(String centerId)`** Returns a String array of all fixes in a center.

AirportInterface API

No.	Type	Method and Description
1	`Airport`	**`select_airport(String airport_code)`** Get an Airport object instance by a given airport code.
2	`String`	**`getArrivalAirport(String acid)`** Get the arrival airport of the requested aircraft.
3	`String`	**`getDepartureAirport(String acid)`** Get the departure airport for the requested aircraft.
4	`double[]`	**`getLocation(String airport_code)`** Get the latitude and longitude of the requested airport. Return an array containing the latitude and longitude.
5	`String`	**`getClosestAirport(double latitude, double longitude)`** Get the code of the airport closest to the given position.
6	`String[]`	**`getAirportsWithinMiles(double lat_deg, double lon_deg, double miles)`** Get all the airports within "miles" range of the given latitude-longitude location.
7	`String`	**`getFullName(String airportid)`** Get the full name corresponding to the given airport code.
8	`Object[]`	**`getAllRunways(String airport_code)`** Get all the runways at a given airport. The returned data is an array. Each element of the array consists of: - Runway name - Waypoint ID
9	`String[]`	**`getRunwayExits(String airport_code, String runway_id)`** Get all the exits at a given runway ID, at a given airport code

10	`Object[]`	**`getLayout_node_map(String airport_code)`** Get the mapping of nodes and the sequence numbers of the surface traffic network at a given airport. The returned data is an array. Each array element consists of: - Waypoint node ID - Node sequence number
11	`Object[]`	**`getLayout_node_data(String airport_code)`** Get the waypoint node data at a given airport. The returned data is an array. Each array element consists of: - Node sequence number - Latitude - Longitude
12	`Object[]`	**`getLayout_links(String airport_code)`** Get links joining the waypoint nodes representing ground layout (runways, taxiways, ramps, and gates) of a given airport which represents the connection of routes between them. The returned data is an array. Each array element consists of: - Node 1 sequence number - Node 2 sequence number
13	`String[]`	**`getSurface_taxi_plan(String acid, String airport_code)`** Get the surface taxi plan of a given aircraft ID at an airport code. Returns an array of all the waypoint IDs in sequential order.
14	`int`	**`generate_surface_taxi_plan(String acid, String airport_code, String startNode_waypoint_id, String endNode_waypoint_id, String runway_name)`** Generate taxi plan and load it in NATS. The function arguments are: acid: Aircraft ID airport_code: Airport code startNode_waypoint_id: Starting waypoint ID endNode_waypoint_id: Ending waypoint ID runway_name: Name of runway Important Note: This function does need the users to specify the V2 for departing aircraft or the touchdown point for arriving aircraft. Return value: 0 means success. 1 means error.
15	`int`	**`setUser_defined_surface_taxi_plan(String acid, String airport_code, String[] user_defined_waypoint_ids)`** Set user-defined surface taxi plan and load it into NATS. Return value:

		0 means success. 1 means error.
16	`String[]`	**`get_taxi_route_from_A_To_B(String acid, String airport_code, String startNode_waypoint_id, String endNode_waypoint_id)`** Generate a taxi route from waypoint A to the waypoint B. Note that this function only returns an array of waypoint IDs.
17	`String`	**`getDepartureRunway(String acid)`** Get the departure runway of the given aircraft. If a departure taxi plan does not exist for the aircraft, no result will be returned.
18	`String`	**`getArrivalRunway(String acid)`** Get the arrival runway of the given aircraft. If an arrival taxi plan does not exist, no result will be returned.
19	`double`	**`getTaxi_tas_knots(String acid)`** Get the surface taxi speed of the given aircraft, knots.
20	`void`	**`setTaxi_tas_knots(String acid, double tas_knots)`** Set the surface taxi speed of the given aircraft, knots.
21	`String[]`	**`getAllAirportCodesInNATS()`** Get ICAO codes for all 57 airports modeled in NATS.
22	`String[]`	**`getRunwayEnds(String airportId, String runwayId)`** Get runway end node waypoints for given airport.

Airport Instance API

No.	Type	Method and Description
1	`String`	**`getCode()`** Get the airport code.
2	`float`	**`getElevation()`** Get the elevation of the airport in feet.
3	`float`	**`getLatitude()`** Get the latitude of the airport.
4	`float`	**`getLongitude()`** Get the longitude of the airport.
5	`String`	**`getName()`** Get the full name of the airport.

TerminalAreaInterface API

No.	Type	Method and Description
1	`String[]`	**`getAllApproaches(String airport_code)`** Get all the Approach Procedures available at the given airport.
2	`String[]`	**`getAllSids(String airport_code)`** Get all the Standard Instrument Departure (SID) Procedures at the given airport.
3	`String[]`	**`getAllStars(String airport_code)`** Get all the Standard Terminal Arrival (STAR) Procedures at the given airport.
4	`String`	**`getCurrentApproach(String acid)`** Get the current Approach Procedure at the given airport for the given flight.
5	`String`	**`getCurrentSid(String acid)`** Get the current SID Procedure at the given airport for the given flight.
6	`String`	**`getCurrentStar(String acid)`** Get the current STAR procedure at the given airport for the given aircraft flight.

7	`String[]`	**`getProcedure_leg_names(String proc_type, String proc_name, String airport_code)`** Get the leg names at the given airport code, procedure type and procedure name. The arguments are: `proc_type`: Procedure type. The valid values are limited to "SID", "STAR" and "APPROACH". `proc_name`: Name of the procedure. `airport_code`: Airport code.
8	`String[]`	**`getWaypoints_in_procedure_leg(String proc_type, String proc_name, String airport_code, String proc_leg_name)`** Get the waypoints at the given airport code, procedure type, procedure name and leg name. Arguments: proc_type: Procedure type. The valid values are limited to "SID", "STAR" and "APPROACH". proc_name: Name of the procedure. airport_code: Airport code.111 proc_leg_name: Name of the procedure leg.
9	`double[]`	**`getWaypoint_Latitude_Longitude_deg(String waypoint_name)`** Get the latitude and longitude (in degrees) of a given waypoint.
10	`double`	**`getProcedure_alt_1(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`** Get the alt 1 value at the given airport code, procedure type, procedure name, leg name and waypoint name. Refer to ARINC 424-18 Section 5.30 for details.
11	`double`	**`getProcedure_alt_2(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`** Get the alt 2 value at the given airport code, procedure type, procedure name, leg name and waypoint name. Refer to ARINC 424-18 Section 5.30 for details.
12	`double`	**`getProcedure_speed_limit(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`** Get the speed limit at the given airport code, procedure type, procedure name, leg name and waypoint name. Refer to ARINC 424-18 Section 5.72 for details.
13	`String`	**`getProcedure_alt_desc(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`** Get the altitude description at the given airport code, procedure type, procedure name, leg name and waypoint name. Refer to ARINC 424-18 Section 5.29 for details.
14	`String`	**`getProcedure_speed_limit_desc(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`** Get the speed limit description at the given airport code, procedure type, procedure name, leg name and waypoint name. Refer to ARINC 424-18 Section

		5.261 for details.

TerrainInterface API

No.	Type	Method and Description
1	`double`	**`getElevation(double latDeg, double lonDeg)`** Returns the terrain elevation (in feet above sea level) at the specified latitude and longitude (degrees). Terrain data from USGS is being used for this function. It has a horizontal resolution of 0.001 degree of latitude/longitude, and vertical resolution of 100ft.
2	`double[]`	**`getElevationAreaStats(double minLatDeg, double maxLatDeg, double minLonDeg, double maxLonDeg)`** Returns an array of statistical information calculated from using terrain elevation data for the specified region. minLatDeg: The lower latitude of the rectangular bounding region (degrees) maxLatDeg: The upper latitude of the rectangular bounding region (degrees) minLonDeg: The lower longitude of the rectangular bounding region (degrees) maxLonDeg: The upper longitude of the rectangular bounding region (degrees) Returns { min, max, mean, variance, stddev } (in feet)
3	`double[][]`	**`getElevationMapBounds()`** Returns the minimum and maximum latitude and longitude bounds of the data used to interpolate elevation data.

WeatherInterface API

No.	Type	Method and Description
1	`int`	**`DownloadWeatherFiles()`** Download aviation weather files. Metar, Sigmet, Pirep files will be downloaded to NATS_Server/share/tg/weather directory from NOAA.
2	`float[]`	**`getWind(float timestamp_sec, float latitude_deg, float longitude_deg, float altitude_ft)`** Get wind data. Returned data is an array of float value. The first element is wind_north vector value. The second element is wind_east vector value.
3	`Weather Polygon []`	**`getWeatherPolygons(String ac_id, double lat_deg, double lon_deg, double alt_ft, double nauticalMile_radius)`** Get weather polygons. Returned data is an array of weather polygons. Notice. This function can only be executed during pause status of simulation.

SafetyMetricsInterface API

No.	Type	Method and Description
1	`Object`	**`getFlightsInRange(String aircraftID)`** This function takes-in the reference aircraft callsign as the input.

		It then forms a bounding box around the aircraft within which a potential safety hazard may exist. The aircraft callsigns are filtered to find the ones that lie within this box, +/- 2000 ft in altitude of the reference aircraft. These flights are then analyzed for their position and velocity relative to the reference aircraft, which are then returned to the user. The returned object is in the following format: [[aircraftCallsign, relativeVelocity, altitudeDifference, bearingAngle, distance], […..], …..]
2	`double`	**`getDistanceToRunwayThreshold(String aircraftId)`** For an aircraft in its takeoff or landing phases, this function calculates the distance to the threshold of the runway from the present position.
3	`double`	**`getDistanceToRunwayEnd(String aircraftId)`** For an aircraft in its takeoff or landing phases, this function calculates the distance to the end of the runway from the present position.
4	`double`	**`getVelocityAlignmentWithRunway(String aircraftId, String procedure)`** For an aircraft either in landing or takeoff phases, this function computes the alignment of the velocity vector relative to the runway centerline. The procedure parameter can have values: 1. ARRIVAL, or 2. DEPARTURE
5	`int`	**`getPassengerCount(String aircraftType)`** This function returns the number of passengers occupying a particular aircraft, assuming 100% load factor. This data is available for all the aircraft types in the BADA database.
6	`double`	**`getAircraftCost(String aircraftType)`** This function returns the cost (in millions of US Dollars) for a new aircraft of the aircraft type. This data is available for all the aircraft types in the BADA database.
7	`Object`	**`getFlightsInWakeVortexRange(String refAircraftId, float envelopeStartWidth, float envelopeStartThickness, float envelopeEndWidth, float envelopeEndThickness, float envelopeRange, float envelopeAltitudeDrop)`** This function models a wake vortex hazard envelope to determine wake encounter hazards for trailing flights. The wake generating aircraft is assumed to be located in the center of a rectangular, divergent, descending tube with two wingspan initial breadth and one wingspan thickness. The function takes in the following parameters: refAircraftId: The callsign of aircraft which is producing the wake vortex. envelopeStartWidth: The width (in feet) of the envelope at start of wake. (typically twice the aircraft wingspan) envelopeStartThickness: The Thickness (in feet) of the envelope at start of the wake. (typically one wingspan of the aircraft) envelopeEndWidth: The width (in feet) of the envelope at end of the wake vortex hazard. envelopeEndThickness: The thickness (in feet) of the envelope at end of the

		wake vortex hazard. envelopeRange: Influence range(in miles) of the vortex envelope. (4 to 15 nm, depending on the weight class of the aircraft: Super, Heavy, Large) envelopeAltitudeDrop: Drop (in feet) of the envelope end relative to the wake generating aircraft. Return Object type for this function is: [[aircraftCallsign, relativeVelocity, altitudeDifference, CourseAngle, distance], […..], …..] An illustration on the use of this function is available at NATS_Client/sample/WakeVortexEnvelope.png
8	`int`	**`setAircraftBookValue(float aircraftBookValue)`** Set the book value of the aircraft in million US$. This is specific to the aircraft instance, and not for an aircraft type.
9	`float`	**`getAircraftBookValue()`** Get the book value of the aircraft in million US$. This is specific to the aircraft instance for a flight in simulation, and not for an aircraft type. To get aircraft cost based on manufacturer model, refer to getAircraftCost() function within SafetyMetricsInterface.
10	`int`	**`setCargoWorth(float cargoWorth)`** Set the value of the cargo in the aircraft, in million US$.
11	`float`	**`getCargoWorth()`** Get the value of the cargo in the aircraft, in million US$.
12	`int`	**`setPassengerLoadFactor(float paxLoadFactor)`** Set load factor for (passenger occupancy relative to the total number of seats) in an aircraft instance. paxLoadFactor ranges from 0 to 1, 0 being an empty aircraft and 1 being fully occupied.
13	`float`	**`getPassengerLoadFactor()`** Get load factor for passenger occupancy in an aircraft instance.
14	`int`	**`setTouchdownPointOnRunway(String aircraftId, double latitude, double longitude)`** Set aircraft touch down point on runway for landing. This would override the touchdown point calculated by the simulation.
15	`double[]`	**`getTouchdownPointOnRunway(String aircraftId)`** Get aircraft touch down point on runway for landing.
16	`int`	**`setTakeOffPointOnRunway(String aircraftId, double latitude, double longitude)`** Set aircraft take off point on runway for liftoff. This would override the take off point calculated by the simulation.
17	`double[]`	**`getTakeOffPointOnRunway(String aircraftId)`** Get aircraft take off point on runway for liftoff.
18	`double`	**`getL1Distance(String airportId, String aircraftId1, String aircraftId2)`** Get L1 distance between two aircraft during surface movements if there is a

		point of potential contact between them in their taxi plans. If there is no possibility of aircraft contact, L1 distance is not defined and the function would return -1.
19	`double`	**`getDistanceToPavementEdge(String airportId, String aircraftId)`** Get distance between aircraft current position and the edge of the pavement in the present direction of travel. This can be used to check if an aircraft might potentially run off of the pavement during taxi, take-off, or ramp operations.

EntityInterface API

No.	Type	Method and Description
1	`ControllerInterface`	**`getControllerInterface()`** Returns a reference to the ControllerInterface.
2	`PilotInterface`	**`getPilotInterface()`** Returns a reference to the PilotInterface.
3	`GroundOperatorInterface`	**`getGroundOperatorInterface()`** Returns a reference to the GroundOperatorInterface.

ControllerInterface API

No.	Type	Method and Description
1	`int`	**`setDelayPeriod(String acid, AircraftClearance aircraft_clearance, int seconds)`** Set delay period in seconds, for providing clearance to an aircraft.
2	`int`	**`int setActionRepeat(String aircraftID, String repeatParameter)`** The controller makes the pilot repeat an action, based on the repeatParameter value. The repeatParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
3	`int`	**`int skipFlightPhase(String aircraftID, String flightPhase)`** The controller skips issuing clearance to an aircraft to the next required flight phase. The flightPhase can have any of the Flight Phase Enum Values. Eg. FLIGHT_PHASE_CLIMB_TO_CRUISE_ALTITUDE
4	`int`	**`int setWrongAction(String aircraftID, String originalChangeParameter, String wrongChangeParameter)`** Instead of clearing the aircraft to the value of one parameter, the controller erroneously clears the aircraft to another value. For example, the controller can assign the magnitude of airspeed (170 kts) as course angle (170 degrees) and viceversa.

		These are following pairs of parameters that can be mutually interchanged: 1. AIRSPEED – COURSE 2. FLIGHT_LEVEL – AIRSPEED 3. COURSE – FLIGHT_LEVEL
5	`int`	**`int setActionReversal(String aircraftID, String changeParameter)`** Controller issues clearance to perform reverse of the intended action, by reversing the value of the changeParameter. The changeParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
6	`int`	**`int setPartialAction(String aircraftID, String changeParameter, float originalTarget, float percentage)`** Clears the aircraft to execute only a part of a required action, by providing the original target value of the parameter, and a percentage of its value to be executed. The changeParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
7	`int`	**`int skipChangeAction(String aircraftID, String skipParameter)`** Omits issuing the clearance by the controller, resulting in the pilot continuing to maintain current value for the skipParameter. The skipParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
8	`int`	**`int setActionLag(String aircraftID, String lagParameter, float lagTimeConstant, float percentageError, float parameterTarget)`** Controller issues lagged clearances lagging the aircraft action. Following are the parameters: The lagParameter (Paremeter to be lagged) can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE lagTimeConstant: To be specified in seconds. 10 seconds, for instance. percentageError: Error percentage for the lag. For example, if 95% of the action is to be executed, percentage error would be 0.05. parameterTarget: Original parameter value to be reached.
9	`int`	**`setControllerAbsence(String aircraftID, int timeSteps)`**

		Controller advisories can be absent for a given time period, requiring the aircraft to execute default plans while waiting for the controller to provide updates. Parameter timeSteps denotes number of steps that aircraft would be flying without controller intervention.
10	`Int`	**`releaseAircraftHold(String aircraftID, String approachProcedure, String targetWaypoint)`** The Controller releases the aircraft from the holding pattern and inserts it into the arrival stream. The controller may clear the aircraft to an approach procedure that may be different from the original flight plan, and a waypoint in that approach. This is the waypoint that the aircraft would intercept to begin approach. For releasing hold pattern in phases other than approach, such as en-route or departure, the approachProcedure parameter needs to be ‘’ (Empty String). The aircraft would get out of the hold and head to the targetWaypoint.
11	`void`	**`enableConflictDetectionAndResolution(boolean flag)`** Enable built-in conflict detection and resolution capability in NATS if boolean_flag = TRUE. Disable NATS built-in conflict detection and resolution capability if boolean_flag = FALSE. Log file is generated in NATS_Server/log directory.
12	`void`	**`setCDR_initiation_distance_ft_surface(float distance)`** Set the initiation distance in feet, for Conflict Detection and Resolution of the surface traffic.
13	`void`	**`setCDR_initiation_distance_ft_terminal(float distance)`** Set the initiation distance in feet for Conflict Detection and Resolution for aircraft flying in the terminal area.
14	`void`	**`setCDR_initiation_distance_ft_enroute(float distance)`** Set the initiation distance in feet, for Conflict Detection and Resolution of en-route air traffic.
15	`void`	**`setCDR_separation_distance_ft_surface(float distance)`** Set the required separation distance in feet for Conflict Detection and Resolution on the surface.
16	`void`	**`setCDR_separation_distance_ft_terminal(float distance)`** Set the required separation distance in feet for Conflict Detection and Resolution in the terminal area.
17	`void`	**`setCDR_separation_distance_ft_enroute(float distance)`** Set the required separation distance in feet for Conflict Detection and Resolution in the en-route airspace.
18	`void`	**`EnableStrategicWeatherAvoidance()`** Enable/disable the strategic weather avoidance capability during simulation. If enabled, the NATS engine checks if any of the flight plans traverse through the adverse weather zone, and creates alternate routes to avoid it. However, if an alternative route is not possible, the aircraft will be held at its current location.

		The strategic weather avoidance logic is executed on an hourly basis. If enabled, NATS simulation will experience significant rise in system resource usage. The simulation will also require higher amounts of execution time.
19	`void`	**`setWeather_polygonFile(String pathFilename)`** Manually set the severe weather polygon file used in strategic weather avoidance. If this function is not used during simulation, NATS engine will choose the latest file. If pathFilename is an empty string "", NATS engine will choose the latest file. If pathFilename is "NONE", polygon file will be disabled.
20	`void`	**`setWeather_sigmetFile(String pathFilename)`** Manually set sigmet file for strategic weather avoidance. If this function is not used during simulation, NATS engine will choose the latest available file. If pathFilename is an empty string "", NATS engine will choose the latest file. If pathFilename is "NONE", sigmet file will be disabled.
21	`int`	**`setTacticalWeatherAvoidance(String waypoint_name, float duration_sec)`** Set waypoint name and duration seconds for weather avoidance. These waypoints are considered to be influenced by the weather so they will be avoided. For setting multiple weather waypoints to avoid, call this function in each waypoint name.
22	`void`	**`enableMergingAndSpacingAtMeterFix(String airportId, String meterFix, String trailAttribute, float timeInTrail/distanceInTrail)`** Enable merging and spacing at a meter fix waypoint on the arrival stream of aircraft. This helps to space out flights for safety reasons both in air and on ground. The function takes in the following parameters: 1. airportId: The ICAO code for the airport. 2. meterFix: The meter fix point where the spacing needs to be enabled. 3. trailAttribute: String, with permitted values being "TIME" or "DISTANCE". This defines whether the float input for the last parameter is distance or time for aircraft spacing. 4. timeInTrail/distanceInTrail: The minimum separation distance or time between aircraft. This input should be consistent with the selection for trailAttribute parameter. timeInTrails is to be supplied in minutes, and distanceInTrail is to be supplied in miles.
23	`void`	**`disableMergingAndSpacingAtMeterFix(String airportId, String meterFix)`** Enable merging and spacing at a meter fix waypoint on the arrival stream of aircraft. This helps to space out flights for safety reasons both in air and on ground. The function takes in the following parameters: 1. airportId: The ICAO code for the airport. 2. meterFix: The meter fix point where the spacing needs to be enabled.

24	`Object[][]`	**`getCDR_status()`** Get current status of CD&R conflicting events Result data: An array of CD&R status. Each array element is formated in the form of an array. The content are: aircraft ID of the held aircraft, aircraft ID of the conflicting aircraft, seconds of holding of the held aircraft Format type: [[String, String, float]] Example: [["AC1", "AC_conflicting_with_AC1", heldSeconds_AC1], ["AC2", "AC_conflicting_with_AC2", heldSeconds_AC2]]

PilotInterface API

No.	Type	Method and Description
1	`int`	**`int setActionRepeat(String aircraftID, String repeatParameter)`** Repeat pilot action, based on the repeatParameter value. The repeatParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
2	`int`	**`int skipFlightPhase(String aircraftID, String flightPhase)`** Ignore the required flight phase transition,. The flightPhase parameter can have any of the Flight Phase Enum Values. Eg. FLIGHT_PHASE_CLIMB_TO_CRUISE_ALTITUDE
3	`int`	**`int setWrongAction(String aircraftID, String originalChangeParameter, String wrongChangeParameter)`** Erroneously set the value of a parameter to another. For example, the pilot can set magnitude of the airspeed (170 kts) as course angle (170 degrees). The following pairs of parameters can be mutually interchanged: 1. AIRSPEED – COURSE 2. FLIGHT_LEVEL – AIRSPEED 3. COURSE – FLIGHT_LEVEL
4	`int`	**`int setActionReversal(String aircraftID, String changeParameter)`** Reverse a pilot action, by reversing the value of changeParameter. changeParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
5	`int`	**`int setPartialAction(String aircraftID, String changeParameter, float originalTarget, float percentage)`** Execute only a part of an action, by providing the original target value of the parameter, and percentage of it to be performed by pilot, for the changeParameter. The changeParameter can have following values:

		1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
6	int	**`int skipChangeAction(String aircraftID, String skipParameter)`** Omit a parameter change by continuing to maintain the current value for the skipParameter. The skipParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE
7	int	**`int setActionLag(String aircraftID, String lagParameter, float lagTimeConstant, float percentageError, float parameterTarget)`** Lag in pilot action, by specifying a certain percent of the execution to be completed within a given time period. Following are the parameters: The lagParameter can have following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE lagTimeConstant: To be specified in seconds. 10 seconds, as an example. percentageError: Error percentage for the lag. For example, if 95% of the action is to be executed in the lag time constant, percentage error would be 0.05. parameterTarget: Original parameter value to be reached.
8	int	**`int setFlightPlanReadError(String aircraftID, String errorParameter, float correctValue)`** If the simulation has not started, the flight plan read from the TRX file can be changed using this function. This constitutes an error in entering the flight plan into the flight management system. Following are the parameters: errorParameter: Parameter with erroneous data. It can have any of the following values: 1. AIRSPEED 2. VERTICAL_SPEED 3. COURSE correctValue: This is the data according to the flight that should have been read.

GroundOperator Interface API

No.	Type	Method and Description
1	int	**`setGroundOperatorAbsence(String groundVehicleId, int timeSteps)`** Ground operators can be absent for a given time period, requiring the vehicle to stop while waiting for the operator to take back control. groundVehicleId: The callsign of the vehicle that the operator is in-charge of.

		timeSteps: Number of time steps for which operator is absent
2	`int`	**`setActionRepeat(String groundVehicleId, String repeatParameter)`** The ground operator repeats an action, based on the repeatParameter value. groundVehicleId: The callsign of the aircraft repeatParameter: Ground vehicle parameter for which action is to be repeated
3	`int`	**`setVehicleContact(String groundVehicleId)`** Ground operators collides the ground vehicle into another object (Potentially building/aircraft/automobile/person) groundVehicleId: The callsign of the vehicle that the operator is in-charge of.
4	`int`	**`setWrongAction(String groundVehicleId, String originalChangeParameter, String wrongChangeParameter)`** Instead of acting to change value of one parameter, the ground operator erroneously changes another. groundVehicleId: The callsign of the ground vehicle originalChangeParameter: Original parameter to be changed due to ground operator action wrongChangeParameter: Erroneous parameter to be changed due to ground operator action
5	`int`	**`setActionReversal(String groundVehicleId, String changeParameter)`** Ground operator executes part of the originally intended action. groundVehicleId: The callsign of the ground vehicle changeParameter: Ground Vehicle parameter for which action is to be partially performed originalTarget: Original value for parameter percentage Percentage of action to be executed
6	`int`	**`setPartialAction(String groundVehicleId, String changeParameter, float originalTarget, float percentage)`** Ground operator executes part of the originally intended action. groundVehicleId: The callsign of the ground vehicle changeParameter: Ground Vehicle parameter for which action is to be partially performed originalTarget: Original value for parameter percentage: Percentage of action to be executed
7	`int`	**`setActionLag(String groundVehicleId, String lagParameter, float lagTimeConstant, float percentageError, float parameterTarget)`** Ground operator lags vehicle action, therreby a certain percent of the execution getting completed within a given time period. groundVehicleId The callsign of the ground vehicle lagParameter: Flight parameter for which action is to be lagged lagTimeConstant: To be specified in seconds. 10 seconds, as an example.

		percentageError: Error percentage for the lag. For example, if 95% of the action is to be executed in the lag time constant, percentage error would be 0.05. parameterTarget: Original parameter value to be reached.

WeatherPolygon Instance API

No.	Type	Method and Description
1	`double[]`	**`getX_data()`** Get longitude values of vertices in the polygon.
2	`double[]`	**`getY_data()`** Get latitude values of vertices in the polygon.
3	`int`	**`getNum_vertices()`** Get number of vertices in the polygon.
4	`boolean`	**`getCcw_flag()`** Get boolean value indicating whether the vertices are created counter-clockwise in the polygon.
5	`double`	**`getXmin()`** Get minimum longitude value of all vertices in the polygon.
6	`double`	**`getXmax()`** Get maximum longitude value of all vertices in the polygon.
7	`double`	**`getYmin()`** Get minimum latitude value of all vertices in the polygon.
	`double`	**`getYmax()`** Get maximum latitude value of all vertices in the polygon.
	`double`	**`getX_centroid()`** Get longitude value of the centroid point in the polygon.
	`double`	**`getY_centroid()`** Get latitude value of the centroid point in the polygon.
	`String`	**`getPoly_type()`** Get polygon type.
	`int`	**`getStart_hr()`** Get starting hour of the polygon.
	`int`	**`getEnd_hr()`** Get ending hour of the polygon.

AircraftClearance Enum Values

Values
AIRCRAFT_CLEARANCE_PUSHBACK
AIRCRAFT_CLEARANCE_TAXI_DEPARTING
AIRCRAFT_CLEARANCE_TAKEOFF
AIRCRAFT_CLEARANCE_ENTER_ARTC
AIRCRAFT_CLEARANCE_DESCENT_FROM_CRUISE
AIRCRAFT_CLEARANCE_ENTER_TRACON
AIRCRAFT_CLEARANCE_APPROACH
AIRCRAFT_CLEARANCE_TOUCHDOWN
AIRCRAFT_CLEARANCE_TAXI_LANDING
AIRCRAFT_CLEARANCE_RAMP_LANDING

Detailed Descriptions of Functions
NATS Client API

Function: `getEntityInterface()`
Return Type: `EntityInterface`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
entityInterface = natsClient.getEntityInterface()
```

Function: `getEnvironmentInterface()`
Return Type: `EnvironmentInterface`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
environmentInterface = natsClient.getEnvironmentInterface()
```

Function: `getEquipmentInterface()`
Return Type: `EquipmentInterface`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
equipmentInterface = natsClient.getEquipmentInterface()
```

Function: `getSafetyMetricsInterface()`
Return Type: `SafetyMetricsInterface`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
```

Function: `getSafetyMInterface()`
Return Type: `SafetyMetricsInterface`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
safetyMInterface = natsClient.getSafetyMInterface()
```

Function: `getSimulationInterface()`
Return Type: `SimulationInterface`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
simulationInterface = natsClient. GetSimulationInterface()
```

Function: `disConnect()`
Return Type: `void`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
natsClient.disConnect()
```

Function: `login(String authenticationID)`
Return Type: `void`
Example:
```
NATSClientFactory = JClass('NATSClientFactory')
natsClient = NATSClientFactory.getNATSClient()
natsClient.login(“ABCD1234”)
```

SimulationInterface API

Function: `clear_trajectory()`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.clear_trajectory()
```

Function: `get_curr_sim_time()`
Return Type: `float`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
currentTime = simulationInterface.get_curr_sim_time()
```

Function: `get_sim_id()`
Return Type: `long`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulation_id = simulationInterface.get_sim_id()
```

Function: `get_runtime_sim_status()`
Return Type: `int`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
currentRuntimeStatus = simulationInterface.get_runtime_sim_status()
```

Function: `pause()`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.pause()
```

Function: `resume()`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.resume()
```

Function: `resume(long timeDuration)`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.resume(1000)
```

Function: `resume(float timeDuration)`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.resume(1000.5)
```

Function: `setupSimulation(long propagationTime, long timeStep)`
Return Type: `int`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.setupSimulation (10000, 5)
```

Function: `setupSimulation(float propagationTime, float timeStep)`
Return Type: `int`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.setupSimulation (100.7, 15.5)
```

Function: `setupSimulation(long propagationTime, long timeStep, long terminalTimeStep, long airborneTimeStep)`
Return Type: `int`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.setupSimulation (1000, 3, 4, 5)
```

Function: `setupSimulation(float propagationTime, float timeStep, float terminalTimeStep, float airborneTimeStep)`
Return Type: `int`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.setupSimulation (1000.0, 3.5, 7.5, 10.3)
```

Function: `start()`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.start()
```

Function: `start(long timeDuration)`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.start(1200)
```

Function: `start(float timeDuration)`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.start(150.65)
```

Function: `startRealTime()`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.startRealTime()
```

Function: `startRealTime_singleUser()`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.startRealTime_singleUser()
```

Function: `stop()`
Return Type: `void`
Example: `simulationInterface = natsClient.getSimulationInterface()`
```
simulationInterface.stop()
```

Function: `write_trajectories(String outputFile)`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.write_trajectories (“SimulationTrajectory.csv”)
```

Function: `request_aircraft(String ac_id)`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.request_aircraft(“ABC123”)
```

Function: `request_groundVehicle(String gv_id)`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.request_groundVehicle(“BUS123”)
```

Function:
```
externalAircraft_create_trajectory_profile(
               String ac_id,
               String ac_type,
               String origin_airport,
               String destination_airport,
               float cruise_altitude_ft,
               float cruise_tas_knots,
               double latitude_deg,
               double longitude_deg,
               double altitude_ft,
               double rocd_fps,
               double tas_knots,
               double course_deg,
               String flight_phase)
```
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.externalAircraft_create_trajectory_profile(
“ABC173”,"B733", "KPHX",
“KSFO”, 33000.0, 430.0, 37.2, -122.4, 2500.0, 215.0, 240.0, 318.2,
“FLIGHT_PHASE_CRUISE")
```

Function:
```
externalAircraft_inject_trajectory_state_data(String ac_id,
double latitude_deg, double longitude_deg,
double altitude_ft, double rocd_fps,
double tas_knots, double course_deg, String flight_phase,
long timestamp_utc_millisec)
```
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.externalAircraft_inject_trajectory_state_data("AB
C123", 32.61, -122.39, 3200,
30, 250, 50, “FLIGHT_PHASE_CRUISE”, 1541784961725)
```

Function: `requestDownloadTrajectoryFile()`
Return Type: `void`
Example:
```
simulationInterface = natsClient.getSimulationInterface()
simulationInterface.requestDownloadTrajectoryFile()
```

EquipmentInterface API

Function: `getAircraftInterface()`
Return Type: `AircraftInterface`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
```

Function: `getGroundVehicleInterface()`
Return Type: `GroundVehicleInterface`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getGroundVehicleInterface ()
```

Function: `getCNSInterface()`
Return Type: `CNSInterface`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getCNSInterface()
```

Function: `getBADADataInterface()`
Return Type: `CNSInterface`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
BADADataInterface = equipmentInterface.getBADADataInterface()
```

AircraftInterface API

Function: `load_aircraft(String trx_file, String mfl_file)`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraftInterface.load_aircraft("share/tg/trx/TRX_DEMO_SFO_PHX_GateTo
Gate.trx", "share/tg/trx/TRX_DEMO_SFO_PHX_mfl.trx")
```

Function: `validate_flight_plan_record(String string_track, String string_fp_route, int mfl_ft)`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
result = aircraftInterface.validate_flight_plan_record("TRACK SWA1897
B733 373628.6 1222248.0 0 0.13 280 ZOA ZOA46", "FP_ROUTE
KSFO./.RW01R.SSTIK4.LOSHN..BOILE..BLH.HYDRR1.I07R.RW07R.<>.KPHX",
37000)
```

Function: `release_aircraft()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraftInterface.release_aircraft()
```

Function: `getAircraftIds(float minLatitude, float maxLatitude, float minLongitude, float maxLongitude, float minAltitude_ft, float maxAltitude_ft)`
Return Type: `String[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraftsIds = aircraftInterface.getAircraftId(28.5, 30.7, 72.8, 74.9, 15000.0, 20000.9)
```

Function: `getAllAircraftId()`
Return Type: `String[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraftsIds = aircraftInterface.getAllAircraftId()
```

Function: `select_aircraft(String aircraft_id)`
Return Type: `Aircraft (Aircraft Instance API)`
Example: `equipmentInterface = natsClient.getEquipmentInterface()`
```
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
```

Function: `synchronize_aircraft_to_server(Aircraft aircraft)`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
synchronize_aircraft_to_server(aircraft)
```

AircraftInstance API

Function: `delay_departure(int delayTimeSeconds)`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.delay_departure(20)
```

Function: `getAcid()`
Return Type: `String`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
aircraftId = aircraft.getAcid()
```

Function: `getAltitude_ft()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
aircraftAltitude = aircraft.getAltitude_ft ()
```

Function: `getCruise_alt_ft()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
aircraftCruiseAltitude = aircraft.getCruise_alt_ft()
```

Function: `getCruise_tas_knots()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
aircraftCruiseAirspeed = aircraft.getCruise_tas_knots()
```

Function: `getDeparture_time_sec()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
flightDepartureTime = aircraft.getDeparture_time_sec()
```
Function: `getDestination_airport_elevation_ft()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
destinationAirportElevation =
aircraft.getDestination_airport_elevation_ft()
```

Function: `getFlight_phase()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightPhase = aircraft.getFlight_phase()
```

Function: `getFlight_plan_latitude_array()`
Return Type: `float[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightLatitudeArray = aircraft.getFlight_plan_latitude_array()
```

Function: `getFlight_plan_length()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightPlanLength = aircraft.getFlight_plan_length()
```

Function: `getFlight_plan_longitude_array()`
Return Type: `float[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightLongitudeArray = aircraft.getFlight_plan_longitude_array()
```

Function: `getFlight_plan_waypoint_name_array()`
Return Type: `String[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightWaypointNameArray = aircraft.getFlight_plan_waypoint_name_array()
```
Function: `getFlight_plan_alt_desc_array()`
Return Type: `String[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightAltitudeDescriptionArray = aircraft.getFlight_plan_alt_desc_array()
```

Function: `getFlight_plan_alt_1_array()`
Return Type: `double[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
flightPlanAltitude1Array = aircraft.getFlight_plan_alt_1_array()
```

Function: `getFlight_plan_alt_2_array()`
Return Type: `double[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
flightPlanAltitude2Array = aircraft.getFlight_plan_alt_2_array()
```

Function: `getFlight_plan_speed_limit_array()`
Return Type: `double[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
flightPlanSpeedLimitArray = aircraft.getFlight_plan_speed_limit_array()
```

Function: `getFlight_plan_speed_limit_desc_array()`
Return Type: `String[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
flightSpeedLimitDescriptionArray =
aircraft.getFlight_plan_speed_limit_desc_array()
```

Function: `getFpa_rad()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
flightPathAngle = aircraft.getFpa_rad()
```

Function: `getCourse_rad()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft(‘ULI-SFD235’)
courseAngle = aircraft.getCourse_rad()
```

Function: `getLanded_flag()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightLandedFlag = aircraft.getLanded_flag()
```

Function: `getLatitude_deg()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightCurrentLatitude = aircraft.getLatitude_deg()
```

Function: `getLongitude_deg()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
flightCurrentLongitude= aircraft.getLongitude_deg()
```

Function: `getOrigin_airport_elevation_ft()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
originAirportElevation = aircraft.getOrigin_airport_elevation_ft()
```

Function: `getRocd_fps()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
rateOfClimbOrDescent = aircraft.getRocd_fps()
```

Function: `getSector_index()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
sectorIndex = aircraft.getSector_index()
```

Function: `getTarget_altitude_ft()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
targetAltitude = aircraft.getTarget_altitude_ft()
```

Function: `getTarget_waypoint_index()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
targetWaypointIndex = aircraft.getTarget_waypoint_index()
```

Function: `getTarget_waypoint_name()`
Return Type: `String`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
targetWaypointName = aircraft.getTarget_waypoint_name()
```

Function: `getTas_knots()`
Return Type: `float`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
currentAirspeed = aircraft.getTas_knots()
```

Function: `getToc_index()`
Return Type: `int`
Example: `equipmentInterface = natsClient.getEquipmentInterface()`
```
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
topOfClimbIndex = aircraft.getToc_index()
```

Function: `getTod_index()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
topOfDescentIndex = aircraft.getTod_index()
```

Function: `setAltitude_ft(float altitude_ft)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setAltitude_ft(27500.8)
```

Function: `setCruise_alt_ft(float cruise_alt_ft)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setCruise_alt_ft(35000.7)
```

Function: `setCruise_tas_knots(float cruise_tas_knots)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setCruise_tas_knots(455.5)
```

Function: `setFlight_phase(int flight_phase)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setFlight_phase(2)
```

Function: `setFlight_plan_latitude_deg(int index, float latitude_deg)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setFlight_plan_latitude_deg(5, 34.50)
```

Function: `setFlight_plan_longitude_deg(int index, float longitude_deg)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setFlight_plan_longitude_deg(5, -122.63)
```

Function: `setLatitude_deg(float latitude_deg)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setLatitude_deg(26.58)
```

Function: `setLongitude_deg(float longitude_deg)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setLongitude_deg (-122.36)
```

Function: `setRocd_fps(float rocd_fps)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setRocd_fps(-50.1)
```

Function: `setTarget_altitude_ft(float target_altitude_ft)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setTarget_altitude_ft(35000.5)
```

Function: `setTarget_waypoint_latitude_deg(float latitude_deg)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setTarget_waypoint_latitude_deg(35.63)
```

Function: `setTarget_waypoint_longitude_deg(float longitude_deg)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setTarget_waypoint_longitude_deg(-118.25)
```

Function: `setTas_knots(float tas_knots)`
Return Type: `void`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
aircraftInterface = equipmentInterface.getAircraftInterface()
aircraft = aircraftInterface.select_aircraft('ULI-SFD235')
aircraft.setTas_knots(400)
```

GroundVehicleInterface API

Function: `load_groundVehicle(String trx_file)`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicleInterface.load_aircraft('share/tg/trx/TRX_GroundVehicles
.trx')
```

Function: `release_groundVehicle()`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicleInterface.release_groundVehicle()
```

Function: `getAssignedGroundVehicleIds()`
Return Type: `String[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
assignedGroundVehicles =
groundVehicleInterface.getAssignedGroundVehicleIds()
```

Function: `getAssignedGroundVehicleIds(String username)`
Return Type: `String[]`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
assignedGroundVehicles =
groundVehicleInterface.getAssignedGroundVehicleIds(username)
```

Function: `getAllGroundVehicleIds()`
Return Type: `String[]`

Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
listGroundVehicle = groundVehicleInterface.getAllGroundVehicleIds()
```

Function: `select_groundVehicle(String groundVehicleId),`
Return Type: `GroundVehicle`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle('BUS123')
```

Function: `externalGroundVehicle_create_trajectory_profile(String groundVehicleId, String aircraftInService, String airport, float latitude, float longitude, float speed, float course)`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicleInterface.groundVehicleInterface.externalGroundVehicle_c
reate_trajectory_profile('NEW123', 'DWA1897', 'KSFO', 37, -122, 15,
28)
```

Function: `externalGroundVehicle_inject_trajectory_state_data(String groundVehicleId, String aircraftInService, float latitude, float longitude, float speed, float course)`
Return Type: `int`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicleInterface.externalGroundVehicle_inject_trajectory_state_
data('NEW123', 'DWA1897', 37, -122, 15, 28)
```

GroundVehicleInstance API

Function: `getGvid()`
Return Type: `String`
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleId = groundVehicle.getGvid()
```

Function: `getAirportId()`

Return Type: String
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleAirportId = groundVehicle.getAirportId()
```

Function: `getAircraftInService()`
Return Type: String
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
aircraftInService = groundVehicle.getAircraftInService()
```

Function: `getFlag_external_groundvehicle()`
Return Type: Boolean,
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
isExternalGroundVehicle =
groundVehicle.getFlag_external_groundvehicle()
```

Function: `getAssigned_user()`
Return Type: String
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
user = groundVehicle.getAssigned_user()
```

Function: `getLatitude()`
Return Type: float
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
latitude = groundVehicle.getLatitude()
```

Function: `setLatitude(float latitude)`

Return Type: void
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicle.setLatitude(37.8959)
```

Function: getLongitude()
Return Type: float
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
longitude = groundVehicle.getLongitude()
```

Function: setLongitude(float longitude)
Return Type: void
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicle.setLongitude(-112.8594)
```

Function: getAltitude()
Return Type: float
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
altitude = groundVehicle.getAltitude()
```

Function: getSpeed()
Return Type: float
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleSpeed = groundVehicle.getSpeed()
```

Function: setSpeed(float speed)

Return Type: void,
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicle.setSpeed(25)
```

Function: getCourse()
Return Type: float
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleCourse = groundVehicle.getCourse()
```

Function: setCourse(float course)
Return Type: void,
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicle.setCourse(1.5)
```

Function: getDeparture_time()
Return Type: float
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleDepartureTime = groundVehicle.getDeparture_time()
```

Function: getDrive_plan_latitude_array()
Return Type: float[]
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleDrivePlanLatitudeArray =
groundVehicle.getDrive_plan_latitude_array()
```

Function: getDrive_plan_longitude_array()

Return Type: float[]
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleDrivePlanLongitudeArray =
groundVehicle.getDrive_plan_longitude_array()
```

Function: getDrive_plan_length()
Return Type: int
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleDrivePlanLength = groundVehicle.getDrive_plan_length()
```

Function: getDrive_plan_waypoint_name_array()
Return Type: String[]
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleDrivePlanWaypointNames =
groundVehicle.getDrive_plan_waypoint_name_array()
```

Function: getTarget_waypoint_index()
Return Type: int
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleTargetWaypointIndex =
groundVehicle.getTarget_waypoint_index()
```

Function: getTarget_waypoint_name()
Return Type: String
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicleTargetWaypointName =
groundVehicle.getTarget_waypoint_name()
```

Function: setDrive_plan_latitude(int index, float latitude)

Return Type: void
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicle.setDrive_plan_latitude(2, 37.2518)
```

Function: setDrive_plan_longitude(int index, float longitude)
Return Type: void
Example:
```
equipmentInterface = natsClient.getEquipmentInterface()
groundVehicleInterface =
equipmentInterface.getGroundVehicleInterface()
groundVehicle = groundVehicleInterface.select_groundVehicle(‘BUS123’)
groundVehicle.setDrive_plan_longitude(2, -112.8155)
```

CNSInterface API

Function: getLineOfSight(double observerLat, double observerLon, double observerAlt, double targetLat, double targetLon, double targetAlt)
Return Type: double[]
Example:
```
cnsInterface = equipmentInterface.getCNSInterface()
cnsInterface.getLineOfSight(33.440903, -111.992862, 1135, 33.274183,
-112.147879, 1500)
```

Function: setNavigationLocationError(String aircraftId, String parameter, double bias, double drift, double scaleFactor, double noiseVariance, int scope)
Return Type: int
Example:
```
cnsInterface = equipmentInterface.getCNSInterface()
cnsInterface.setNavigationLocationError('SWA1897', 'LATITUDE',
0.00005, 0.00000001, 0.9, 0.2, 1)
cnsInterface.setNavigationLocationError('SWA1897', 'LONGITUDE',
0.00005, 0.00000001, 0.9, 0.2, 1)
```

Function: setNavigationAltitudeError(String aircraftId, double bias, double noiseVariance, int scope)
Return Type: int
Example:
```
cnsInterface = equipmentInterface.getCNSInterface()
cnsInterface.setNavigationAltitudeError('SWA1897', .00005, 0.2, 0)
```

Function: `setRadarError(String airportId, String parameter, double originalValue, double bias, double noiseVariance, int scope)`
Return Type: `int`
Example:
```
cnsInterface = equipmentInterface.getCNSInterface()
cnsInterface.setRadarError('KSFO', 'RANGE', 25, 0.0000005, 0.2, 1)
cnsInterface.setRadarError('KSFO', 'AZIMUTH', 30, 0.0000005, 0.2, 1)
cnsInterface.setRadarError('KSFO', 'ELEVATION', 2500,0.0000005,0.2,1)
```

BADADataInterface API

Function: `getBADA_cruiseTas(String ac_type, double altitude_ft)`
Return Type: `double`
Example:
```
badaDataInterface = equipmentInterface.getBADADataInterface()
badaDataInterface.getBADA_cruiseTas('B733', 15000)
```

Function: `getBADA_climbRate_fpm(String ac_type, double flt_level, String bada_mass)`
Return Type: `double`
Example:
```
badaDataInterface = equipmentInterface.getBADADataInterface()
badaDataInterface.getBADA_climbRate_fpm('B733', 150, 'NOMINAL')
```

Function: `getBADA_climbTas(String ac_type, double altitude_ft)`
Return Type: `double`
Example:
```
badaDataInterface = equipmentInterface.getBADADataInterface()
badaDataInterface.getBADA_climbTas('B733', 15000)
```

Function: `getBADA_descentRate_fpm(String ac_type, double flight_level, String bada_mass)`
Return Type: `double`
Example:
```
badaDataInterface = equipmentInterface.getBADADataInterface()
badaDataInterface.getBADA_descentRate_fpm('B733', 150, 'NOMINAL')
```

Function: `getBADA_descentTas(String ac_type, double altitude_ft)`
Return Type: `double`
Example:
```
badaDataInterface = equipmentInterface.getBADADataInterface()
badaDataInterface.getBADA_descentTas('B733', 15000)
```

EnvironmentInterface API

Function: `load_rap(String windDirectory)`
Return Type: `void`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
environmentInterface.load_rap("share/tg/rap")
```

Function: `release_rap()`
Return Type: `int`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
environmentInterface.release_rap()
```

Function: `getAirportInterface()`
Return Type: `AirportInterface`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
```

Function: `getTerrainInterface()`
Return Type: `TerrainInterface`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terrainInterface = environmentInterface.getTerrainInterface()
```

Function: `getTerminalAreaInterface()`
Return Type: `TerminalAreaInterface`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface = environmentInterface.getTerminalAreaInterface()
```

Function: `getWeatherInterface()`
Return Type: `WeatherInterface`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
weatherInterface = environmentInterface.getWeatherInterface()
```

Function: `getCenterCodes()`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
environmentInterface.getCenterCodes()
```

Function: `getCurrentCenter(String aircraftId)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
environmentInterface.getCurrentCenter('SWA1897')
```

Function: `getFixesInCenter(String centerId)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
environmentInterface.getFixesInCenter('KZOA')
```

AirportInterface API

Function: `select_airport(String airport_code)`
Return Type: `Airport`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airport = airportInterface.select_airport(“KPHX”)
```

Function: `getArrivalAirport(String acid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
arrivalAirport = airportInterface.getArrivalAirport(‘ULI-SFD235’)
```

Function: `getDepartureAirport(String acid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
departureAirport = airportInterface.getDepartureAirport(‘ULI-SFD235’)
```

Function: `getLocation(String airport_code)`
Return Type: `double[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportLocation = airportInterface.getLocation(‘KLAX’)
```

Function: `getClosestAirport(double latitude, double longitude)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
closestAirport = airportInterface.getClosestAirport(35.2, -118.6)
```

Function: `getAirportsWithinMiles(double lat_deg, double lon_deg, double miles)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airports = airportInterface.getAirportsWithinMiles(35.2, -118.6,
22.5)
```

Function: `getFullName(String airportid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportFullName = airportInterface.getFullName('KJFK')
```

Function: `getAllRunways(String airport_code)`
Return Type: `Object[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportRunways = airportInterface.getAllRunways('PANC')
```

Function: `getAllGates(String airport_code)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportGates = airportInterface.getAllGates('PANC')
```

Function: `getRunwayExits(String airport_code, String runway_id)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
runwayExits = airportInterface.getRunwayExits('KSFO', 'RW28R')
```

Function: `getLayout_node_map(String airport_code)`
Return Type: `Object[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportLayoutNodeMap = airportInterface.getLayout_node_map('PHNL')
```

Function: `getLayout_node_data(String airport_code)`
Return Type: `Object[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportLayoutNodeData = airportInterface .getLayout_node_data('PHNL')
```

Function: `getLayout_links(String airport_code)`
Return Type: `Object[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportLayoutLinks = airportInterface.getLayout_links('PHNL')
```

Function: `getSurface_taxi_plan(String acid, String airport_code)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
surfaceTaxiPlan = airportInterface.getSurface_taxi_plan('ULI-SFD235', 'KSFO')
```

Function: `generate_surface_taxi_plan(String acid, String airport_code, String startNode_waypoint_id, String endNode_waypoint_id, String runway_name)`
Return Type: `int`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
generatedTaxiPlan = airportInterface.generate_surface_taxi_plan('ULI-
SFD235', 'KSFO',
'Gate_01_001', 'Rwy_02_001', 'RW06L')
```

Function: `setUser_defined_surface_taxi_plan(String acid, String airport_code, String[] user_defined_waypoint_ids)`
Return Type: `int`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
generatedTaxiPlan =
airportInterface.setUser_defined_surface_taxi_plan('ULI-SFD235',
'KSFO',
['Gate_01_001', 'Ramp_01_001', 'Txy_01_001', 'Txy_01_002',
'Rwy_02_001'])
```

Function: `get_taxi_route_from_A_To_B(String acid, String airport_code, String startNode_waypoint_id, String endNode_waypoint_id)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
taxiPlanAtoB = airportInterface.get_taxi_route_from_A_To_B('ULI-
SFD235', 'KSFO', 'Gate_01_001', 'Rwy_02_001')
```

Function: `getDepartureRunway(String acid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
departureRunway = airportInterface.getDepartureRunway('ULI-SFD235').
```

Function: `getArrivalRunway(String acid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
arrivalRunway = airportInterface.getArrivalRunway(‘ULI-SFD235’)
```

Function: `getTaxi_tas_knots(String acid)`
Return Type: `double`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
taxiSpeed = airportInterface.getTaxi_tas_knots(‘ULI-SFD235’)
```

Function: `setTaxi_tas_knots(String acid, double tas_knots)`
Return Type: `void`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportInterface.setTaxi_tas_knots(‘ULI-SFD235’, 25.0)
```

Function: `getAllAirportCodesInNATS()`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportList = airportInterface.getAllAirportCodesInNATS()
```

Function: `getRunwayEnds(String airportId, String runwayId)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airportList = airportInterface.getrunwayEnds(“KSFO”, “RW28R”)
```

AirportInstance API

Function: `getCode()`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airport = airportInterface.select_airport(“KORD”)
airportCode = airport.getCode()
```

Function: `getElevation()`
Return Type: `float`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airport = airportInterface.select_airport("KORD")
airportElevation = airport.getElevation()
```

Function: `getLatitude()`
Return Type: `float`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airport = airportInterface.select_airport("KORD")
airportLatitude = airport.getLatitude()
```

Function: `getLongitude()`
Return Type: `float`
Example:
```
airportInterface = environmentInterface.getAirportInterface()
airport = airportInterface.select_airport("KORD")
airportLongitude = airport.getLongitude()
```

Function: `getName()`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
airportInterface = environmentInterface.getAirportInterface()
airport = airportInterface.select_airport("KORD")
airportName = airport.getName()
```

TerminalAreaInterface API

Function: `getAllApproaches(String airport_code)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
approaches = terminalAreaInterface.getAllApproaches('KORD')
```

Function: `getAllSids(String airport_code)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
sids = terminalAreaInterface.getAllSids('KORD')
```

Function: `getAllStars(String airport_code)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
stars = terminalAreaInterface.getAllStars('KORD')
```

Function: `getCurrentApproach(String acid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
currentApproach = terminalAreaInterface.getCurrentApproach('ULI-
SFD235')
```

Function: `getCurrentSid(String acid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
currentSid = terminalAreaInterface.getCurrentSid('ULI-SFD235')
```

Function: `getCurrentStar(String acid)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
currentStar = terminalAreaInterface.getCurrentStar('ULI-SFD235')
```

Function: `getProcedure_leg_names(String proc_type, String proc_name, String airport_code)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
sidLegNames = terminalAreaInterface.getProcedure_leg_names("SID",
"SSTIK3", "KSFO")
```

Function: `getWaypoints_in_procedure_leg(String proc_type, String proc_name, String airport_code,String proc_leg_name)`
Return Type: `String[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
waypointNames = terminalAreaInterface.getWaypoints_in_procedure_leg(“SID”,
“SSTIK3”, “KSFO”,
“PORTE”)
```

Function: `getClosestWaypoint(float[][] waypointOptions, float[] targetWaypoint)`
Return Type: `int`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
closestWaypointIndex =
terminalAreaInterface.getClosestWaypoint([[37.61,-122.3],[42.9,-
75.61]], [43.9,-77.6])
```

Function: `calculateWaypointDistance(float latx, float lonx, float laty, float lony)`
Return Type: `double`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
waypointDistance =
terminalAreaInterface.calculateWaypointDistance(37.61,-122.3,42.9,-
75.61)
```

Function: `getWaypoint_Latitude_Longitude_deg(String waypoint_name)`
Return Type: `double[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
waypointLocation =
terminalAreaInterface.getWaypoint_Latitude_Longitude_deg(‘BOILE’)
```

Function: `getProcedure_alt_1(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`
Return Type: `double`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
procedureAlt1 = terminalAreaInterface.getProcedure_alt_1(“SID”,
“SSTIK3”, “KSFO”, “PORTE”,
“KAYEX”)
```

Function: `getProcedure_alt_2(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`
Return Type: `double`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
procedureAlt2 = terminalAreaInterface.getProcedure_alt_2(“SID”,
“SSTIK3”, “KSFO”, “PORTE”, “KAYEX”)
```

Function: `getProcedure_speed_limit(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`
Return Type: `double`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
procedureSpeedLimit =
terminalAreaInterface.getProcedure_speed_limit(“SID”, “SSTIK3”,
“KSFO”, “PORTE”, “KAYEX”)
```

Function: `getProcedure_alt_desc(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
procedureAltitudeDesc =
terminalAreaInterface.getProcedure_alt_desc(“SID”, “SSTIK3”, “KSFO”,
“PORTE”, “KAYEX”)
```

Function: `getProcedure_speed_limit_desc(String proc_type, String proc_name, String airport_code, String proc_leg_name, String proc_wp_name)`
Return Type: `String`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terminalAreaInterface =
environmentInterface.getTerminalAreaInterface()
procedureSpeedLimitDesc =
terminalAreaInterface.getProcedure_speed_limit_desc (“SID”, “SSTIK3”,
“KSFO”, “PORTE”, “KAYEX”)
```

TerrainInterface API

Function: `getElevation(double latDeg, double lonDeg)`
Return Type: `double`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terrainAreaInterface = environmentInterface.getTerrainInterface()
elevation = terrainAreaInterface.getElevation(34.5, -122.23)
```

Function: `getElevationAreaStats(double minLatDeg, double maxLatDeg, double minLonDeg, double maxLonDeg)`
Return Type: `double[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terrainAreaInterface = environmentInterface.getTerrainInterface()
elevationAreaStats = terrainAreaInterface.getElevationAreaStats(34.5,
-122.23, 36.8, -121.9)
```

Function: `getElevationMapBounds()`
Return Type: `double[][]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
terrainAreaInterface = environmentInterface.getTerrainInterface()
elevationMapBounds = terrainAreaInterface.getElevationMapBounds()
```

EntityInterface API

Function: `getControllerInterface()`
Return Type: `ControllerInterface`
Example:
```
entityInterface = natsClient.getEntityInterface()
controllerInterface = entityInterface.getControllerInterface()
```

Function: `getPilotInterface()`
Return Type: `PilotInterface`
Example:
```
entityInterface = natsClient.getEntityInterface()
pilotInterface = entityInterface.getPilotInterface()
```

Function: `getGroundOperatorInterface()`
Return Type: `GroundOperatorInterface`
Example:
```
entityInterface = natsClient.getEntityInterface()
groundOperatorInterface = entityInterface.getGroundOperatorInterface ()
```

WeatherInterface API

Function: `DownloadWeatherFiles()`
Return Type: `int`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
weatherInterface = environmentInterface.getWeatherInterface()
weatherInterface.DownloadWeatherFiles()
```

Function:
```
getWind(float timestamp_sec,
        float latitude_deg,
        float longitude_deg,
        float altitude_ft)
```
Return Type: `float[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
weatherInterface = environmentInterface.getWeatherInterface()
windValue = weatherInterface.getWind(6600.0, 40.0, -73.0, 20000.0)
```

Function: `getWeatherPolygons(String ac_id, double lat_deg, double lon_deg, double alt_ft, double nauticalMile_radius)`
Return Type: `WeatherPolygon[]`
Example:
```
environmentInterface = natsClient.getEnvironmentInterface()
weatherInterface = environmentInterface.getWeatherInterface()
windValue = weatherInterface.getWeatherPolygons("UA123", 48.0,
-120.0, 33000.0, 100.0)
```

ControllerInterface API

Function: `setDelayPeriod(String acid, AircraftClearance aircraft_clearance, int seconds)`
Return Type: `int`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
```

```
setDelayPeriod = controllerInterface.setDelayPeriod(‘ULI-SFD235’,
AIRCRAFT_CLEARANCE_TAXI_DEPARTING, 10)
```

Function: `setActionRepeat(String aircraftID, String repeatParameter)`
Return Type: `int`
Example:

```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setActionRepeat('ULI-SFD235', 'COURSE')
```

Function: `skipFlightPhase(String aircraftID, String flightPhase)`
Return Type: `int`
Example:

```
controllerInterface = entityInterface.getControllerInterface()
controllerInterfaceskipFlightPhase('ULI-SFD235',
'FLIGHT_PHASE_CLIMB_TO_CRUISE_ALTITUDE')
```

Function: `setWrongAction(String aircraftID, String originalChangeParameter, String wrongChangeParameter)`
Return Type: `int`
Example:

```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setWrongAction('ULI-SFD235', 'COURSE',
'AIRSPEED')
```

Function: `setActionReversal(String aircraftID, String changeParameter)`
Return Type: `int`
Example:

```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setActionReversal('ULI-SFD235', 'COURSE')
```

Function: `setPartialAction(String aircraftID, String changeParameter, float originalTarget, float percentage)`
Return Type: `int`
Example:

```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setPartialAction('ULI-SFD235', 'VERTICAL_SPEED',
200, 25)
```

Function: `skipChangeAction(String aircraftID, String skipParameter)`
Return Type: `int`
Example:

```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.skipChangeAction('ULI-SFD235', 'COURSE')
```

Function: `setActionLag(String aircraftID, String lagParameter, float lagTimeConstant, float percentageError, float parameterTarget)`

Return Type: `int`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setActionLag('ULI-SFD235', 'COURSE', 10,0.05, 30)
```

Function: `setControllerAbsence(string aircraftID, int timeSteps)`
Return Type: `int`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setControllerAbsence ('ULI-SFD235', 5)
```

Function: `releaseAircraftHold(String aircraftID, String approach, String targetWaypoint)`
Return Type: `int`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.releaseAircraftHold('ULI-SFD235', 'I07L',
'FFIXA')
```
Function: `enableConflictDetectionAndResolution(boolean flag)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.enableConflictDetectionAndResolution(True)
```

Function: `setCDR_initiation_distance_ft_surface(float distance)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setCDR_initiation_distance_ft_surface(50000.0)
```

Function: `setCDR_initiation_distance_ft_terminal(float distance)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setCDR_initiation_distance_ft_terminal(50000.0)
```

Function: `setCDR_initiation_distance_ft_enroute(float distance)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setCDR_initiation_distance_ft_enroute(50000.0)
```

Function: `setCDR_separation_distance_ft_surface(float distance)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setCDR_separation_distance_ft_surface(50000.0)
```

Function: `setCDR_separation_distance_ft_terminal(float distance)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setCDR_separation_distance_resolve_ft_terminal(50
000.0)
```

Function: `setCDR_separation_distance_resolve_ft_enroute(float distance)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setCDR_separation_distance_ft_enroute(50000.0)
```

Function: `enableStrategicWeatherAvoidance()`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.enableStrategicWeatherAvoidance()
```
Function: `setWeather_polygonFile(String pathFilename)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setWeather_polygonFile(“share/rg/polygons/xxxx.da
t”)
```

Function: `setWeather_sigmetFile(String pathFilename)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.setWeather_sigmetFile(“share/tg/weather/xxxx.sigm
et”)
```

Function: `setTacticalWeatherAvoidance(String waypoint_name, float duration_sec)`
Return Type: `int`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
flag = controllerInterface.setTacticalWeatherAvoidance(“ABCDE”, 100)
```

Function: `enableMergingAndSpacingAtMeterFix(String airportId, String meterFix, String trailAttribute, float timeInTrail/distanceInTrail)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.enableMergingAndSpacingAtMeterFix(“KPHX”,
“GEELA”, “DISTANCE”, 4.5)
```

Function: `disableMergingAndSpacingAtMeterFix(String airportId, String meterFix)`
Return Type: `void`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
controllerInterface.enableMergingAndSpacingAtMeterFix(“KPHX”, “GEELA”)
```

Function: `getCDR_status()`
Return Type: `Object[][]`
Example:
```
controllerInterface = entityInterface.getControllerInterface()
cdrStatus = controllerInterface.getCDR_status()
```

SafetyMetricsInterface API

Function: `getFlightsInRange(String aircraftID)`
Return Type: `Object`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
flightsInRange = safetyMetricsInterface.getFlightsInRange('ULI-SFD235')
```

Function: `getDistanceToRunwayThreshold(String aircraftID)`
Return Type: `double`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
distance = safetyMetricsInterface.getDistanceToRunwayThreshold ('ULI-SFD235')
```

Function: `getDistanceToRunwayEnd(String aircraftID)`
Return Type: `double`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
distance = safetyMetricsInterface. getDistanceToRunwayEnd ('ULI-SFD235')
```

Function: `getVelocityAlignmentWithRunway(String aircraftID, String procedure)`
Return Type: `double`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
alignmentAngle = safetyMetricsInterface.
GetVelocityAlignmentWithRunway ('ULI-SFD235', ‘DEPARTURE’)
```

Function: `getPassengerCount(String aircraftType)`
Return Type: `int`

Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
passengerCount = safetyMetricsInterface. getPassengerCount ('A306')
```

Function: `getAircraftCost(String aircraftID)`
Return Type: `double`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
flightsInRange = safetyMetricsInterface.getAircraftCost ('A306')
```

Function: `getFlightsInWakeVortexRange(String refAircraftId, float envelopeStartLength, float envelopeStartBreadth, float envelopeEndLength, float envelopeEndBreadth, float envelopeRange, float envelopeAltitudeDrop)`
Return Type: `Object`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.getFlightsInWakeVortexRange('SWA1897', 200,
150, 400, 350, 2, 50)
```
Function: `setAircraftBookValue(String aircraftId, float aircraftBookValue)`
Return Type: `int`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.setAircraftBookValue(‘SWA1897’, 5.6)
```

Function: `setCargoWorth(String aircraftId, float cargoWorth)`
Return Type: `int`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.setCargoWorth(‘SWA1897’, 1.2)
```

Function: `setPassengerLoadFactor(String aircraftId, float paxLoadFactor)`
Return Type: `int`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.setPassengerLoadFactor(‘SWA1897’, 0.72)
```

Function: `getAircraftBookValue(String aircraftId)`
Return Type: `float`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
aircraftBookValue =
safetyMetricsInterface.getAircraftBookValue(‘SWA1897’)
```

Function: `getCargoWorth(String aircraftId)`
Return Type: `float`

Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
cargoWorth = safetyMetricsInterface.getCargoWorth('SWA1897')
```

Function: `getPassengerLoadFactor(String aircraftId)`
Return Type: `float`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
paxLoadFactor =
safetyMetricsInterface.getPassengerLoadFactor('SWA1897')
```

Function: `setTouchdownPointOnRunway(String aircraftId, float latitude, float longitude)`
Return Type: `float`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.setTouchdownPointOnRunway('SWA1897', 32.423,
-123.123)
```

Function: `getTouchdownPointOnRunway(String aircraftId)`
Return Type: `float`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
print safetyMetricsInterface.getTouchdownPointOnRunway('SWA1897')
```

Function: `setTakeOffPointOnRunway(String aircraftId, float latitude, float longitude)`
Return Type: `float`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.setTakeOffPointOnRunway('SWA1897', 37.625735,
-122.368191)
```

Function: `getTakeOffPointOnRunway(String aircraftId)`
Return Type: `float`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
print safetyMetricsInterface.getTakeOffPointOnRunway('SWA1897')
```

Function: `getL1Distance(String airportId, String aircraftId1, String aircraftId2)`
Return Type: `double`
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.getL1Distance('KSFO', 'SWA1897', 'SWA1898')
```

Function: `getDistanceToPavementEdge(String airportId, String aircraftId)`

Return Type: double
Example:
```
safetyMetricsInterface = natsClient.getSafetyMetricsInterface()
safetyMetricsInterface.getDistanceToPavementEdge(‘KSFO’, ‘SWA1897’)
```

PilotInterface API

Function: `setActionRepeat(String aircraftID, String repeatParameter)`
Return Type: int
Example:
```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.setActionRepeat('ULI-SFD235', 'COURSE')
```

Function: `skipFlightPhase(String aircraftID, String flightPhase)`
Return Type: int
Example:
```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.skipFlightPhase('ULI-SFD235',
'FLIGHT_PHASE_CLIMB_TO_CRUISE_ALTITUDE')
```
Function: `setWrongAction(String aircraftID, String originalChangeParameter, String wrongChangeParameter)`
Return Type: int
Example:
```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.setWrongAction('ULI-SFD235', 'COURSE', 'AIRSPEED');
```

Function: `setActionReversal(String aircraftID, String changeParameter)`
Return Type: int
Example:
```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.setActionReversal('ULI-SFD235', 'COURSE')
```
Function: `setPartialAction(String aircraftID, String changeParameter, float originalTarget, float percentage)`
Return Type: int
Example:
```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.setPartialAction('PLEASE_ENTER_AIRCRAFT_CALLSIGN_HERE'
, 'VERTICAL_SPEED', 200, 25);
```

Function: `skipChangeAction(String aircraftID, String skipParameter)`
Return Type: int
Example:
```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.skipChangeAction('ULI-SFD235', 'COURSE')
```

Function: `setActionLag(String aircraftID, String lagParameter, float lagTimeConstant, float percentageError, float parameterTarget)`

Return Type: `int`
Example:

```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.setActionLag('ULI-SFD235', 'COURSE', 10, 0.05, 30)
```

Function: `setFlightPlanReadError(String aircraftID, String errorParameter, float updatedValue)`
Return Type: `int`
Example:

```
pilotInterface = entityInterface.getPilotInterface()
pilotInterface.setFlightPlanReadError('ULI-SFD235', 'VERTICAL_SPEED',
398.0)
```

GroundOperatorInterface API

Function: `setGroundOperatorAbsence(String groundVehicleId, int timeSteps)`
Return Type: `int`
Example:

```
groundOperatorInterface =
entityInterface.getGroundOperatorInterface()
groundOperatorInterface.setGroundOperatorAbsence('BUS123', 4)
```

Function: `setActionRepeat(String groundVehicleId, String repeatParameter)`
Return Type: `int`
Example:

```
groundOperatorInterface =
entityInterface.getGroundOperatorInterface()
groundOperatorInterface.setActionRepeat('BUS123', 'SPEED')
```

Function: `setVehicleContact(String groundVehicleId)`
Return Type: `int`
`Interface:GroundOperatorInterface`
Example:

```
groundOperatorInterface =
entityInterface.getGroundOperatorInterface()
groundOperatorInterface.setVehicleContact('BUS123')
```

Function: `setActionReversal(String groundVehicleId, String changeParameter)`
Return Type: `int`
Example:

```
groundOperatorInterface =
entityInterface.getGroundOperatorInterface()
groundOperatorInterface.setActionReversal('BUS123', 'COURSE')
```

Function: `setPartialAction(String groundVehicleId, String changeParameter, float originalTarget, float percentage),`
Return Type: `int`
Example:
```
groundOperatorInterface =
entityInterface.getGroundOperatorInterface()
groundOperatorInterface.setPartialAction('BUS123', 'SPEED', 8, 50)
```

Function: `setActionLag(String groundVehicleId, String lagParameter, float lagTimeConstant, float percentageError, float parameterTarget)`
Return Type: `int`
Example:
```
groundOperatorInterface =
entityInterface.getGroundOperatorInterface()
groundOperatorInterface.setActionLag('BUS123', 'SPEED', 10, 0.5, 30)
```

WeatherPolygon API

Function: `getX_data()`
Return Type: `double[]`
Example:
```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
x_data_array = weatherPolygons[0].getX_data()
```

Function: `getY_data()`
Return Type: `double[]`
Example:
```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
y_data_array = weatherPolygons[0].getY_data()
```

Function: `getNum_vertices()`
Return Type: `int`
Example:
```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getNum_vertices()
```

Function: `getCcw_flag()`
Return Type: `boolean`
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getCcw_flag()
```

Function: getXmin()
Return Type: double
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getXmin()
```

Function: getXmax()
Return Type: double
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getXmax()
```

Function: getYmin()
Return Type: double
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getYmin()
```

Function: getYmax()
Return Type: double
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getYmax()
```

Function: getX_centroid()
Return Type: double
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getX_centroid()
```

Function: getY_centroid()
Return Type: double
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
```

```
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getY_centroid()
```

Function: `getPoly_type()`
Return Type: `String`
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getPoly_type()
```

Function: `getStart_hour()`
Return Type: `int`
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getStart_hour()
```

Function: `getEnd_hour()`
Return Type: `int`
Example:

```
weatherInterface = environmentInterface.getWeatherInterface()
weatherPolygons = weatherInterface.getWeatherPolygons('UA123', 48.0,
-120.0, 33000.0, 100.0)
weatherPolygons[0].getEnd_hour()
```