

---

# **pydcs Documentation**

***Release 0.8***

**Peinthor Rene**

March 26, 2016



<b>1</b>	<b>Quickstart</b>	<b>3</b>
<b>2</b>	<b>Random mission script</b>	<b>5</b>
<b>3</b>	<b>dcx.mission module</b>	<b>7</b>
<b>4</b>	<b>dcx.task module</b>	<b>19</b>
<b>5</b>	<b>dcx</b>	<b>35</b>
5.1	dcx package . . . . .	35
<b>6</b>	<b>Indices and tables</b>	<b>63</b>
	<b>Python Module Index</b>	<b>65</b>



pydcs is a Python(3) framework for creating digital combat simulator missions.

Contents:



---

### Quickstart

---

Creating a mission is fairly simple, the most important class in this respect is `dcsmission.Mission`. This class contains all information for running a dcs mission. It is a .zip file that contains several lua data structures and other resources like briefing images, voice overs and other lua scripts.

```
m = dcs.Mission()
m.save('mission.miz')
```

This code is enough to create a mission file without any unit groups in the Caucasus(default) terrain.

To add a A-10C flight group starting from Batumi airport use the following snippet:

```
fg = m.flight_group_from_airport(m.country("USA"), "A-10C Flight Group",
                                dcs.planes.A-10C, m.terrain.batumi(), group_size=2)
fg.units[0].set_player()
```

This adds a A-10C flight with 2 planes starting cold from a free parking slot. In the next line it also sets the first unit of the flight as player. For more options when adding a flight see `dcsmission.Mission.flight_group_from_airport()`.





---

## Random mission script

---

pydcs delivers a small random mission creator script. This script was indented to show features of the framework and is also a little testbed.

It is invoked by the commandline with the following arguments:

```
usage: random_mission.py [-h] [-a {A-10C,Su-25T,M-2000C,Ka-50,MiG-21Bis}]
                        [-p PLAYERCOUNT] [-s {inflight,runway,warm,cold}]
                        [-t {main,CAS,CAP,refuel}]
                        [-d {random,day,night,dusk,dawn,noon}]
                        [-w {dynamic,dyncyclone,dynanti,dynone,clear}] [-u]
                        [--show-stats] [-o OUTPUT]

Random DCS mission generator

optional arguments:
  -h, --help                show this help message and exit
  -a {A-10C,Su-25T,M-2000C,Ka-50,MiG-21Bis}, --aircrafttype {A-10C,Su-25T,M-2000C,Ka-50,MiG-21Bis}
                                Player aircraft type
  -p PLAYERCOUNT, --playercount PLAYERCOUNT
                                Player count
  -s {inflight,runway,warm,cold}, --start {inflight,runway,warm,cold}
                                Start mission type
  -t {main,CAS,CAP,refuel}, --missiontype {main,CAS,CAP,refuel}
                                Mission type
  -d {random,day,night,dusk,dawn,noon}, --daytime {random,day,night,dusk,dawn,noon}
                                Daytime
  -w {dynamic,dyncyclone,dynanti,dynone,clear}, --weather {dynamic,dyncyclone,dynanti,dynone,clear}
                                Weather
  -u, --unhide              Show enemy pre mission
  --show-stats              Show generated missions stats
  -o OUTPUT, --output OUTPUT
                                Name and path of the generated mission
```



---

## dcx.mission module

---

The mission module is the entry point to all pydcx functions.

**class** `dcx.mission.StartType`

Bases: `enum.Enum`

Enum class for start types.

**Cold = None**

Coldstart from ramp.

**Warm = None**

Warmstart from ramp.

**Runway = None**

Start from runway.

**static from\_string** (*s: str*)

Returns the StartType enum for a string value.

["cold", "warm", "runway"]

**Parameters** *s* – string representation of the starttype

**Returns** the correct StartType.

**class** `dcx.mission.Mission` (*terrain: typing.Union=None*)

Bases: `object`

This class represents the whole dcx .miz file.

A .miz file is a zip file containing all needed files to run a mission. example.miz:

- mission

- options

- warehouses

- i10n

- DEFAULT

- dictionary

- \*mapResource

- \*[localized resource files, .wav, .jpg, ...]

**Parameters** *terrain* – the used terrain for this mission.

**season\_from\_start\_time** = None

If set to True the mission season will be set by the value of `Mission.start_time`

**load\_file** (*filename: str*)

Load a mission file (.miz) file, replacing all current data.

**Parameters** **filename** – path to the mission(.miz) file.

**Returns** True if everything loaded correctly

**Return type** bool

**Raises** `RuntimeError` – if an unknown value is encountered

**sortie\_text** () → str

Returns the mission sortie text.

**Returns** the mission sortie text

**set\_sortie\_text** (*text: str*)

Sets the mission sortie text.

**Parameters** **text** – text to set.

**description\_text** () → str

Returns the mission description text.

**Returns** the mission description text

**set\_description\_text** (*text: str*)

Sets the mission description text.

**Parameters** **text** – text to set.

**description\_bluetask\_text** () → str

Returns the blue task description text.

**Returns** the blue task description text

**set\_description\_bluetask\_text** (*text: str*)

Sets the red coalitions task description text.

**Parameters** **text** – text to set.

**description\_redtask\_text** () → str

Returns the red task description text.

**Returns** the red task description text

**set\_description\_redtask\_text** (*text: str*)

Sets the red coalitions task description text.

**Parameters** **text** – text to set.

**add\_picture\_red** (*filepath: str*) → str

Adds a new briefing picture to the red coalition.

**Parameters** **filepath** – path to the image, jpg or bmp.

**Returns** the resource key of the picture

**add\_picture\_blue** (*filepath: str*) → str

Adds a new briefing picture to the blue coalition.

**Parameters** **filepath** – path to the image, jpg or bmp.

**Returns** the resource key of the picture

**next\_group\_id()**

Get the next free group id

**Returns** a new group id

**next\_unit\_id()**

Get the next free unit id

**Returns** a new unit id

**next\_dict\_id()**

Get the next free dictionary id

**Returns** a new dictionary id

**eplrs\_for**(*group: str*) → typing.Dict

Searches all vehicle eplrs using groups and writes them in a mapping

**Parameters** **group** – which group to look for eplrs task, ["helicopter", "plane", "vehicle"]

**Returns** a dict mapping groups to used eplrs id

**next\_eplrs**(*group\_type: str*) → int

Get next eplrs for the given group type.

**Parameters** **group\_type** – one of "vehicle", "helicopter" or "plane"

**Returns** the next eplrs id to use

**Return type** int

**string**(*s, lang='DEFAULT'*)

Create a new String() object for translation

**Parameters**

- **s** – string for lang
- **lang** – language for s

**Returns** A new String() object for string s

**Return type** *String*

**static**(*name, \_type: dcs.unittype.UnitType*) → dcs.unit.Static

Creates a plain static object to be added to a group

**Parameters**

- **name** – of the static object
- **\_type** (*StaticType*) – type of the static

**Returns** a new static object

**Return type** *Static*

**static\_group**(*country, name, \_type: dcs.unittype.UnitType, position: dcs.mapping.Point, heading=0, hidden=False, dead=False*)

Add a static group with 1 static object.

**Parameters**

- **country** (*Country*) – the object belongs too
- **name** – name of the group
- **\_type** – what kind of object

- **position** (`dcs.mapping.Point`) – where to place the object
- **heading** – of the object
- **hidden** – should the object be hidden on the map
- **dead** – should the object be rendered as dead

**Returns** the new static group

**Return type** *StaticGroup*

**vehicle** (*name*, *\_type*: `dcs.unittypes.VehicleType`) → `dcs.unit.Vehicle`

Creates a plain vehicle unit to be added to a group

**Parameters**

- **name** – of the vehicle
- **\_type** – vehicle type

**Returns** a new vehicle unit.

**Return type** *Vehicle*

**vehicle\_group** (*country*, *name*, *\_type*: `dcs.unittypes.VehicleType`, *position*: `dcs.mapping.Point`, *heading*=0, *group\_size*=1, *formation*=`<Formation.Line: 1>`, *move\_formation*: `dcs.point.PointAction=<PointAction.OffRoad: 'Off Road'>`) → `dcs.unitgroup.VehicleGroup`

Adds a new vehicle group to the given country.

**Parameters** **country** – which the vehicle group will belong too

**Returns** the new vehicle group object

**Return type** *VehicleGroup*

**vehicle\_group\_platoon** (*country*, *name*, *types*: `typing.List`, *position*: `dcs.mapping.Point`, *heading*=0, *formation*=`<Formation.Line: 1>`, *move\_formation*: `dcs.point.PointAction=<PointAction.OffRoad: 'Off Road'>`) → `dcs.unitgroup.VehicleGroup`

Adds a new vehicle group to the given country and given vehicle types.

**Parameters** **country** – which the vehicle group will belong too

**Returns** the new vehicle group object

**Return type** *VehicleGroup*

**ship** (*name*, *\_type*: `dcs.unittypes.ShipType`) → `dcs.unit.Ship`

Creates a plain ship unit to be added to a group

**Parameters**

- **name** – of the ship
- **\_type** – ship type

**Returns** a new ship unit.

**Return type** *Ship*

**ship\_group** (*country*, *name*, *\_type*: `dcs.unittypes.ShipType`, *position*: `dcs.mapping.Point`, *heading*=0, *group\_size*=1) → `dcs.unitgroup.ShipGroup`

Adds a ship group to the given country.

**Parameters**

- **country** (*Country*) – which the ship group will belong too
- **name** – of the ship group
- **\_type** – which kind of ship to add
- **position** (*dcs.mapping.Point*) – where the new group will be placed
- **heading** – initial heading of the group, only used if no additional waypoints
- **group\_size** – how many ships of \_type

**Returns** the new ship group object

**Return type** *ShipGroup*

**plane\_group** (*name*) → *dcs.unitgroup.PlaneGroup*

This creates a plain plane group without any units or starting points.

This method is a advanced interface method not intended for simple use. For adding full featured plane group see

- *flight\_group()*
- *flight\_group\_inflight()*
- *flight\_group\_from\_airport()*

**Parameters** **name** – Group name

**Returns** A new *dcs.unitgroup.PlaneGroup*

**Return type** *PlaneGroup*

**plane** (*name*, *\_type*: *dcs.planes.PlaneType*, *country*: *dcs.country.Country*)

Creates a new plane unit.

This method is a advanced interface method not intended for simple usage. For adding full a featured plane group see

- *flight\_group()*
- *flight\_group\_inflight()*
- *flight\_group\_from\_airport()*

**Parameters**

- **name** – unit name
- **\_type** – type of the plane
- **country** (*Country*) – the plane belongs, needed for default liveries

**Returns** A new *dcs.unit.Plane*

**Return type** *Plane*

**helicopter** (*name*, *\_type*: *dcs.helicopters.HelicopterType*, *country*: *dcs.country.Country*)

Creates a new helicopter unit.

This method is a advanced interface method not intended for simple usage. For adding full a featured helicopter group see

- *flight\_group()*
- *flight\_group\_inflight()*

- `flight_group_from_airport()`

**Parameters**

- **name** – unit name
- **\_type** – type of the helicopter
- **country** (`Country`) – the helicopter belongs, needed for default liveries

**Returns** A new `dcs.unit.Helicopter`**Return type** `Helicopter`**aircraft** (`name`, `_type: dcs.unittypes.FlyingType`, `country: dcs.country.Country`) → `typing.Union`Creates a new plane or helicopter unit, depending on the `_type`.

This method is a advanced interface method not intended for simple usage. For adding full a featured plane/helicopter group see

- `flight_group()`
- `flight_group_inflight()`
- `flight_group_from_airport()`

**Parameters**

- **name** – unit name
- **\_type** – type of the aircraft
- **country** (`Country`) – the aircraft belongs, needed for default liveries

**Returns** A new `dcs.unit.Plane` or `dcs.unit.Helicopter`**Return type** `Helicopter`**helicopter\_group** (`name`) → `dcs.unitgroup.HelicopterGroup`

Creates a plain helicopter group without any units or starting points.

This method is a advanced interface method not intended for simple usage. For adding a full featured helicopter group see

- `flight_group()`
- `flight_group_inflight()`
- `flight_group_from_airport()`

**Parameters** **name** – Group name**Returns** A new `dcs.unitgroup.HelicopterGroup`**Return type** `HelicopterGroup`**flight\_group\_inflight** (`country`, `name: str`, `aircraft_type: dcs.unittypes.FlyingType`, `position: dcs.mapping.Point`, `altitude: int`, `speed=None`, `maintask: typing.Union=None`, `group_size: int=1`) → `typing.Union`

Add a new Plane/Helicopter group inflight.

The type of the resulting group depends on the given `aircraft_type`.

**Parameters**

- **country** (`Country`) – the new group will belong to



- **name** – of the new group
- **aircraft\_type** (*FlyingType*) – type of all units in the group
- **position** (*dcs.mapping.Point*) – where the new group will be placed
- **altitude** – of the new group
- **speed** – of the new group, if none a default will be picked
- **maintask** (*MainTask*) – if none the default task for the aircraft\_type will be used
- **group\_size** – number of units in the group(maximum 4 or 1 for certain types)

Returns a new *dcs.unitgroup.PlaneGroup* or *dcs.unitgroup.HelicopterGroup*

Return type *FlyingGroup*

```
flight_group_from_airport (country:      dcs.country.Country,      name,      aircraft_type:
                             dcs.unittypes.FlyingType,  airport:  dcs.terrain.terrain.Airport,
                             maintask:      dcs.task.MainTask=None,      start_type:
                             dcs.mission.StartType=<StartType.Cold: 1>,  group_size=1,
                             parking_slots: typing.List=None) → typing.Union
```

Add a new Plane/Helicopter group at the given airport.

Runway, warm/cold start depends on the given start\_type.

#### Parameters

- **country** (*Country*) – Country object the plane group belongs to
- **name** – Name of the aircraft group
- **maintask** (*MainTask*) – Task of the aircraft group
- **aircraft\_type** (*FlyingType*) – FlyingType class that describes the aircraft\_type
- **airport** (*Airport*) – Airport object on which to spawn the helicopter
- **start\_type** (*StartType*) – Start from runway, cold or warm parking position
- **parking\_slots** – List of parking slots to use for aircrafts
- **group\_size** – number of units in the group(maximum 4 or 1 for certain types)

Returns a new *dcs.unitgroup.PlaneGroup* or *dcs.unitgroup.HelicopterGroup*

Return type *FlyingGroup*

```
flight_group_from_unit (country:      dcs.country.Country,      name,      aircraft_type:
                             dcs.unittypes.FlyingType,      carrier_unit:      typing.Union,
                             maintask:      dcs.task.MainTask=None,      start_type:
                             dcs.mission.StartType=<StartType.Cold: 1>,  group_size=1) →
                             typing.Union
```

Add a new Plane/Helicopter group at the given FARP or carrier unit.

#### Parameters

- **country** (*Country*) – Country object the plane group belongs to
- **name** – Name of the aircraft group
- **maintask** (*MainTask*) – Task of the aircraft group
- **aircraft\_type** (*FlyingType*) – FlyingType class that describes the aircraft\_type
- **carrier\_unit** (*Unit*) – Group(Ship, FARP) on which to spawn

- **start\_type** (*StartType*) – Start from runway, cold or warm parking position, ignored for now
- **group\_size** – number of units in the group(maximum 4 or 1 for certain types)

**Returns** a new *dcs.unitgroup.PlaneGroup* or *dcs.unitgroup.HelicopterGroup*

**Return type** *FlyingGroup*

**flight\_group** (*country: dcs.country.Country, name: str, aircraft\_type: dcs.unittypes.FlyingType, airport: typing.Union, position: typing.Union, altitude=3000, speed=500, maintask: typing.Union=None, start\_type: dcs.mission.StartType=<StartType.Runway: 3>, group\_size=1*) → *dcs.unitgroup.FlyingGroup*

This is wrapper around *flight\_group\_inflight* and *flight\_group\_from\_airport*.

Depending on the airport parameter a flight group will added inflight or on an airport.

#### Parameters

- **country** (*Country*) – Country object the plane group belongs to
- **name** – Name of the aircraft group
- **aircraft\_type** (*FlyingType*) – *FlyingType* class that describes the aircraft\_type
- **airport** (*Airport*) – Airport object on which to spawn the helicopter
- **position** (*dcs.mapping.Point*) – where the new group will be placed, if inflight
- **altitude** – initial altitude of the group if inflight
- **speed** – initial speed of the group if inflight
- **maintask** (*MainTask*) – Task of the aircraft group
- **start\_type** (*StartType*) – Start from runway, cold or warm parking position
- **group\_size** – number of units in the group(maximum 4 or 1 for certain types)

**Returns** a new *dcs.unitgroup.PlaneGroup* or *dcs.unitgroup.HelicopterGroup*

**Return type** *FlyingGroup*

**awacs\_flight** (*country: dcs.country.Country, name: str, plane\_type: dcs.planes.PlaneType, airport: typing.Union, position: dcs.mapping.Point, race\_distance=30000, heading=90, altitude=4500, speed=550, start\_type: dcs.mission.StartType=<StartType.Cold: 1>, frequency=140*) → *dcs.unitgroup.PlaneGroup*

Add an AWACS flight group.

This is simple way to add an AWACS flight group to your mission. It needs an initial orbit point, race distance and heading from this point.

If an airport is given the AWACS flight will start from there otherwise, it will placed 2 km in front of the reference position.

#### Parameters

- **country** (*Country*) – Country object the awacs group belongs to
- **name** – of the AWACS flight
- **plane\_type** (*PlaneType*) – AWACS plane type. e.g E\_3A
- **airport** (*Airport*) – starting airport, use None if you want it to spawn inflight
- **position** (*dcs.mapping.Point*) – reference point for the race-track
- **race\_distance** – distance for the race-track pattern

- **heading** – direction from the referene position
- **altitude** – of the AWACS race-track
- **speed** – of the AWACS flight
- **start\_type** (*StartType*) – of the flight if starts from airport
- **frequency** – VHF-AM frequency in mhz

**Returns** the created AWACS flight group

**Return type** *PlaneGroup*

**refuel\_flight** (*country*, *name*: *str*, *plane\_type*: *dcs.planes.PlaneType*, *airport*: *typing.Union*, *position*: *dcs.mapping.Point*, *race\_distance*=30000, *heading*=90, *altitude*=4500, *speed*=407, *start\_type*: *dcs.mission.StartType*=<*StartType.Cold*: 1>, *frequency*=140, *tacanchannel*='10X') → *dcs.unitgroup.PlaneGroup*

Add an refuel flight group.

This is simple way to add an refuel flight group to your mission. It needs an initial orbit point, race distance and heading from this point.

If an airport is given the refuel flight will start from there otherwise, it will placed 2 km in front of the reference position.

#### Parameters

- **country** (*Country*) – Country object the awacs group belongs to
- **name** – of the refuel flight
- **plane\_type** (*PlaneType*) – refuel plane type. e.g KC\_135
- **airport** (*Airport*) – starting airport, use None if you want it to spawn inflight
- **position** (*dcs.mapping.Point*) – reference point for the race-track
- **race\_distance** – distance for the race-track pattern
- **heading** – direction from the referene position
- **altitude** – of the refuel race-track
- **speed** – of the refuel flight
- **start\_type** (*StartType*) – of the flight if starts from airport
- **frequency** – VHF-AM frequency in mhz
- **tacanchannel** – if the *PlaneType* supports tacan this channel will be set.

**Returns** the created refuel flight group

**Return type** *PlaneGroup*

**escort\_flight** (*country*, *name*: *str*, *escort\_type*: *dcs.planes.PlaneType*, *airport*: *typing.Union*, *group\_to\_escort*: *dcs.unitgroup.FlyingGroup*, *start\_type*: *dcs.mission.StartType*=<*StartType.Cold*: 1>, *group\_size*=2)

Add an escort flight group to the mission.

An escort flight is a flight group that will use the *dcs.task.EscortTaskAction* to escort another flight group.

If no airport is given, the escort flight will spawn near the group to escort.

#### Parameters

- **country** (*Country*) – the escort flight belongs too

- **name** – of the flight group
- **escort\_type** (*PlaneType*) – *PlaneType* for the escort task
- **airport** (*Airport*) – starting airport, use None if you want it to spawn inflight
- **group\_to\_escort** – id of the group to escort
- **start\_type** (*StartType*) – of the flight if starts from airport
- **group\_size** – how many planes should be in the escort flight

**Returns** the created escort group

**Return type** *PlaneGroup*

**patrol\_flight** (*country*, *name*: str, *patrol\_type*: dcs.planes.PlaneType, *airport*: typing.Union, *pos1*, *pos2*, *start\_type*: dcs.mission.StartType=<StartType.Cold: 1>, *speed*=600, *altitude*=4000, *max\_engage\_distance*=60000, *group\_size*=2)

Add an patrol flight group to the mission.

A patrol flight is a flight group that will fly a orbit between 2 given points and will engage any incoming air threats within *max\_engage\_distance*.

If no airport is given, the patrol flight will spawn near the first patrol point(*pos1*).

#### Parameters

- **country** (*Country*) – the flight belongs too
- **name** – name of the patrol flight
- **patrol\_type** (*PlaneType*) – *PlaneType* for the patrol flight
- **airport** (*Airport*) – starting airport, use None if you want it to spawn inflight
- **pos1** (*dcs.mapping.Point*) – first orbit waypoint
- **pos2** (*dcs.mapping.Point*) – second orbit waypoint
- **start\_type** (*StartType*) – of the flight if starts from airport
- **speed** – orbit speed
- **altitude** – initial altitude and orbit altitude
- **max\_engage\_distance** – the distance in KM the patrol flight will respond to enemy threats
- **group\_size** – how many planes should be in the flight group

**Returns** the created patrol group

**Return type** *PlaneGroup*

**country** (*name*)

Returns the country object for the mission by the given string

**Parameters** **name** – string representation of the country

**Returns** the object of the country, None if not found.

**Return type** *Country*

**find\_group** (*group\_name*, *search*='exact') → typing.Union

Searches a group with the given name.

#### Parameters

- **group\_name** – part or exact name of the group

- **search** – search mode to use
  - ‘exact’: whole name must match
  - ‘match’: part of the name must match

**Returns** the group found, otherwise None

**Return type** *Group*

**is\_red** (*country: dcs.country.Country*) → bool

Checks if the given country object is part o the red coalition.

**Parameters** **country** (*Country*) – object to check

**Returns** True if it is part of the red coalition, else False.

**Return type** bool

**is\_blue** (*country: dcs.country.Country*) → bool

Checks if the given country object is part o the blue coalition.

**Parameters** **country** (*Country*) – object to check

**Returns** True if it is part of the blue coalition, else False.

**Return type** bool

**stats** () → typing.Dict

Gather some mission stats.

This method counts up the different group types and used units and returns them as easy to print dict.

**Returns** dict containing various group and unit counts.

**print\_stats** (*d*)

Print the given mission stats to standard output.

**Parameters** **d** – stats dict to print, *dcs.mission.Mission.stats()*

**reload** ()

Reloads the current loaded file

**Raises** `RuntimeError` – if there is currently no file loaded.

**save** (*filename=None*)

Save the current Mission object to the given file.

**Parameters**

- **filename** – filepath to save the Mission object
- **show\_stats** (*bool*) – if True print mission stats to standard out.

**dict** ()

**class** *dcs.mission.MapResource* (*mission: dcs.mission.Mission*)

Bases: `object`

*MapResource* is responsibly to manage all additional mission resource files.

Mission resource files are briefing images, lua scripts, sounds files.

**Parameters** **mission** (*Mission*) – the mission this *MapResource* belongs too, needed for dictionary ids

**load\_from\_dict** (*\_dict, zipf: zipfile.ZipFile, lang='DEFAULT'*)

**add\_resource\_file** (*filepath*, *lang*=*'DEFAULT'*, *key*=*None*)

Adds a file to the mission resource depot.

**Parameters**

- **filepath** – path to the file to add
- **lang** – language this file belongs too.
- **key** – should None, needed for loading

**Returns** resource key to use in scripts

**store** (*zipf*: *zipfile.ZipFile*, *lang*=*'DEFAULT'*)

**class** `dcs.mission.Options`

Bases: `object`

Should be a representation for the mission options file might be removed in the future.

**load\_from\_dict** (*d*)

---

## dcx.task module

---

This module holds all Tasks that are possible to specify in dcs.

There are 2 type of tasks, a MainTask and a Task action.

- MainTasks are the flight groups main objective like *CAS*, *CAP*, *SEAD*, ...
- Task actions on the otherhand are specific tasks within a MainTask, these cover things like *AttackGroup*, *Orbit*, *Follow*, *Escort*, ...

Also options and commands are task actions.

**class** `dcx.task.Modulation`

Bases: `enum.Enum`

Enum for VHF frequency bands

**AM** = `None`

AM frequency band

**FM** = `None`

FM frequency band

**class** `dcx.task.Task` (*\_id*)

Bases: `object`

Base class for task actions.

**classmethod** `create_from_dict` (*d*)

**dict** ()

**class** `dcx.task.ControlledTask` (*task: dcx.task.Task=None*)

Bases: `dcx.task.Task`

A ControlledTask is a task action with start and stop conditions.

ControlledTask is a wrapper around a normal task action that has special methods to add start/stop conditions.

**Parameters** **task** – to wrap

**Id** = `'ControlledTask'`

**start\_after\_time** (*time: int*)

Start the wrapped task after time seconds.

**Parameters** **time** – start after x seconds.

**start\_if\_user\_flag** (*user\_flag, value: bool*)

Start the wrapped task if user\_flag has value.

**Parameters**

- **user\_flag** – id of the userflag
- **value** – bool value of the flag

**start\_probability** (*probability: int*)

Chance that the wrapped task will actually start.

**Parameters probability** – start chance in %

**start\_if\_lua\_predicate** (*lua\_predicate: str*)

Start wrapped task if lua condition is true.

**Parameters lua\_predicate** – lua condition as string

**stop\_after\_time** (*time: int*)

Stop the wrapped task after time seconds.

**Parameters time** – start after x seconds.

**stop\_if\_user\_flag** (*user\_flag, value: bool*)

Stop the wrapped task if user\_flag has value.

**Parameters**

- **user\_flag** – id of the userflag
- **value** – bool value of the flag

**stop\_if\_lua\_predicate** (*lua\_predicate: str*)

Stop wrapped task if lua condition is true.

**Parameters lua\_predicate** – lua condition as string

**stop\_after\_duration** (*duration: int*)

Stop task after duration seconds.

**Parameters duration** – in seconds

**class** `dcs.task.WeaponType`

Bases: `enum.Enum`

**class** `dcs.task.TargetType`

Bases: `type`

**id** = `None`

**class** `dcs.task.Targets`

Bases: `object`

**class** `All`

Bases: `object`

**id** = `'All'`

**class** `Air`

Bases: `object`

**id** = `'Air'`

**class** `Planes`

Bases: `object`

**id** = `'Planes'`



```
class Fighters
    Bases: object

    id = 'Fighters'

class Targets.All.Air.Planes.Bombers
    Bases: object

    id = 'Bombers'

class Targets.All.Air.Helicopters
    Bases: object

    id = 'Helicopters'

class Targets.All.GroundUnits
    Bases: object

    id = 'Ground Units'

class Infantry
    Bases: object

    id = 'Infantry'

class Targets.All.GroundUnits.Fortifications
    Bases: object

    id = 'Fortifications'

class Targets.All.GroundUnits.GroundVehicles
    Bases: object

    id = 'Ground vehicles'

class ArmoredVehicles
    Bases: object

    id = 'Armored vehicles'

class Tanks
    Bases: object

    id = 'Tanks'

class Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.IFV
    Bases: object

    id = 'IFV'

class Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.APC
    Bases: object

    id = 'APC'

class Targets.All.GroundUnits.GroundVehicles.Artillery
    Bases: object

    id = 'Artillery'

class Targets.All.GroundUnits.GroundVehicles.UnarmedVehicles
    Bases: object

    id = 'Unarmed vehicles'

class Targets.All.GroundUnits.AirDefence
    Bases: object
```

```
    id = 'Air Defence'

    class AAA
        Bases: object

        id = 'AAA'

        class SAMRelated
            Bases: object

            id = 'SAM related'

            class SRSAM
                Bases: object

                id = 'SR SAM'

            class Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.MRSAM
                Bases: object

                id = 'MR SAM'

            class Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.LRSAM
                Bases: object

                id = 'LR SAM'

class Targets.All.Naval
    Bases: object

    id = 'Naval'

    class Ships
        Bases: object

        id = 'Ships'

        class ArmedShips
            Bases: object

            id = 'Armed ships'

            class HeavyArmedShips
                Bases: object

                id = 'Heavy armed ships'

            class AircraftCarriers
                Bases: object

                id = 'Aircraft Carriers'

            class Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Cruisers
                Bases: object

                id = 'Cruisers'

            class Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Destroyers
                Bases: object

                id = 'Destroyers'

            class Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Frigates
                Bases: object

                id = 'Frigates'
```

```
class Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Corvettes
    Bases: object
    id = 'Corvettes'

class Targets.All.Naval.Ships.ArmedShips.LightArmedShips
    Bases: object
    id = 'Light armed ships'

class Targets.All.Naval.Ships.UnarmedShips
    Bases: object
    id = 'Unarmed ships'

class Targets.All.Naval.Submarines
    Bases: object
    id = 'Submarines'

class dcs.task.NoTask
    Bases: dcs.task.Task
    Id = 'NoTask'

class dcs.task.AttackGroup(group_id=0, weapon_type: dcs.task.WeaponType=<WeaponType.Auto: 1073741822>)
    Bases: dcs.task.Task
    Attack group task action
    Parameters
        • group_id – group id to attack
        • weapon_type – WeaponType to use for the attack
    Id = 'AttackGroup'

class dcs.task.AttackUnit(unit_id=0, attack_limit: typing.Union=None, dcs.task.WeaponType=<WeaponType.Auto: 1073741822>, group_attack=False)
    Bases: dcs.task.Task
    Attack unit task action
    Parameters
        • unit_id – unit id to attack
        • attack_limit – how much attack runs
        • weapon_type – WeaponType to use for the attack
        • group_attack – indicate if the unit must be attacked by all aircrafts in the group.
    Id = 'AttackUnit'

class dcs.task.AttackMapObject(position: dcs.mapping.Point=Point(0, 0), attack_limit: typing.Union=None, dcs.task.WeaponType=<WeaponType.Auto: 1073741822>, group_attack=False)
    Bases: dcs.task.Task
    Id = 'AttackMapObject'

class dcs.task.AntishipStrikeTaskAction
    Bases: dcs.task.Task
    Id = 'EngageTargets'
```

```
Key = 'AntiShip'
dict()

class dcs.task.CASTaskAction
    Bases: dcs.task.Task

    Id = 'EngageTargets'
    Key = 'CAS'
    dict()

class dcs.task.SEADTaskAction
    Bases: dcs.task.Task

    Id = 'EngageTargets'
    Key = 'SEAD'
    dict()

class dcs.task.CAPTaskAction
    Bases: dcs.task.Task

    Id = 'EngageTargets'
    Key = 'CAP'
    dict()

class dcs.task.FighterSweepTaskAction
    Bases: dcs.task.Task

    Id = 'EngageTargets'
    Key = 'FighterSweep'
    dict()

class dcs.task.EscortTaskAction(group_id=None, engagement_max_dist=60000, lastwpt=None,
                                targets: typing.List=None, position: typing.Dict=None)
    Bases: dcs.task.Task

    Id = 'Escort'

class dcs.task.Expend
    Bases: enum.Enum

class dcs.task.Bombing(position: dcs.mapping.Point=Point(0, 0), weapon_type:
                        dcs.task.WeaponType=<WeaponType.Auto: 1073741822>, expend:
                        dcs.task.Expend=<Expend.Auto: 'Auto'>, attack_qty=1, group_attack=False,
                        direction: typing.Union=None, altitude: typing.Union=None)
    Bases: dcs.task.Task

    Id = 'Bombing'

class dcs.task.BombingRunway(airport_id: int=0, weapon_type: dcs.task.WeaponType=<WeaponType.Auto:
                              1073741822>, expend: dcs.task.Expend=<Expend.Auto: 'Auto'>,
                              attack_qty=1, group_attack=False, direction: typing.Union=None,
                              altitude: typing.Union=None)
    Bases: dcs.task.Task

    Id = 'BombingRunway'

class dcs.task.EngageTargets(max_distance=None, targets: typing.List=None)
    Bases: dcs.task.Task
```

```
    Id = 'EngageTargets'

class dcs.task.EngageTargetsInZone (position: dcs.mapping.Point=Point(0, 0), radius=5000, tar-
                                gets: typing.List=None)
    Bases: dcs.task.Task
    Id = 'EngageTargetsInZone'

class dcs.task.EngageGroup (group_id=0, visible=False)
    Bases: dcs.task.Task
    Id = 'EngageGroup'

class dcs.task.EngageUnit (unit_id=0, visible=False)
    Bases: dcs.task.Task
    Id = 'EngageUnit'

class dcs.task.FireAtPoint (position: dcs.mapping.Point=Point(0, 0), rounds=None, radius=0)
    Bases: dcs.task.Task
    Id = 'FireAtPoint'

class dcs.task.Hold
    Bases: dcs.task.Task
    Unit will hold current position
    Id = 'Hold'

class dcs.task.AWACSTaskAction
    Bases: dcs.task.Task
    Id = 'AWACS'

class dcs.task.RefuelingTaskAction
    Bases: dcs.task.Task
    Assigns the aircraft group to refuel at the nearest tanker aircraft.
    Id = 'Refueling'

class dcs.task.Tanker
    Bases: dcs.task.Task
    Assigns the aircraft to act as an Airborne tanker.
    Id = 'Tanker'

class dcs.task.OrbitAction (altitude=4000, speed=600, pattern: dcs.task.OrbitPattern=<OrbitPattern.RaceTrack:
                                'Race-Track'>)
    Bases: dcs.task.Task
    Id = 'Orbit'

    class OrbitPattern
        Bases: enum.Enum

class dcs.task.Follow (groupid=None, position: dcs.mapping.Point=Point(-200, 0),
                    altitude_difference=-200, last_wpt=None)
    Bases: dcs.task.Task
    Id = 'Follow'

class dcs.task.Aerobatics
    Bases: dcs.task.Task
    Id = 'Aerobatics'
```

```
class dcs.task.Designation
```

```
    Bases: enum.Enum
```

```
class dcs.task.FAC (callsign: int=1, designation: dcs.task.Designation=<Designation.Auto: 'Auto'>, frequency: int=30, modulation: dcs.task.Modulation=<Modulation.FM: 1>, number: int=1)
```

```
    Bases: dcs.task.Task
```

```
    Id = 'FAC'
```

```
class dcs.task.FACEngageGroup (group_id: int=0, visible=False, weapon_type: dcs.task.WeaponType=<WeaponType.Auto: 1073741822>, priority: int=0, callsign: int=1, designation: dcs.task.Designation=<Designation.Auto: 'Auto'>, frequency: int=30, modulation: dcs.task.Modulation=<Modulation.FM: 1>, datalink=True, number: int=1)
```

```
    Bases: dcs.task.Task
```

```
    Id = 'FAC_EngageGroup'
```

```
class dcs.task.Land (position: dcs.mapping.Point=Point(0, 0), duration: int=None)
```

```
    Bases: dcs.task.Task
```

Helicopter landing task.

If added to a helicopter group, the group will land at the given coordinates for the given duration.

**Parameters**

- **position** – *dcs.mapping.Point* where to land
- **duration** – how long the helicopter should stay on ground in seconds.

```
    Id = 'Land'
```

```
class dcs.task.Embarking (position: dcs.mapping.Point=Point(0, 0), groupids: typing.List=None, distribution: typing.Dict=None, duration: int=None)
```

```
    Bases: dcs.task.Task
```

Pickup task for helicopters.

Helicopter group will land at the given coordinates and will pickup the given groups.

**Parameters**

- **position** – *dcs.mapping.Point* where to land and pickup
- **groupids** – list of groups to pickup
- **distribution** – dictionary with heli unit to groups to pickup mapping {heliunit: [grp1, grp2], ..}
- **duration** – how long the helicopter should stay on ground and wait in seconds.

```
    Id = 'Embarking'
```

```
class dcs.task.EmbarkToTransport (position: dcs.mapping.Point=Point(0, 0), zone_radius=200, concrete_unitid=None)
```

```
    Bases: dcs.task.Task
```

Task for ground units that will get picked up by a helicopter

**Parameters**

- **position** – *dcs.mapping.Point* where to wait to get picked up.
- **zone\_radius** – radius around the point where the group will embark.
- **concrete\_unitid** – if specified the group will embark to exactly this unit.

```
    Id = 'EmbarkToTransport'
```

```
class dcs.task.DisembarkFromTransport (position: dcs.mapping.Point=Point(0, 0),
                                         zone_radius=200)
```

Bases: *dcs.task.Task*

Task for ground units that will disembark a transport helicopter

**Parameters**

- **position** – *dcs.mapping.Point* where the group will disembark
- **zone\_radius** – radius around the point where the group will disembark.

**Id** = 'DisembarkFromTransport'

```
class dcs.task.CargoTransportation (groupid=None, zoneid=None)
```

Bases: *dcs.task.Task*

Task for Cargo transportation.

**Parameters**

- **groupid** – cargo group id
- **zoneid** – zone id to transport to??

**Id** = 'CargoTransportation'

```
class dcs.task.WrappedAction
```

Bases: *dcs.task.Task*

**Id** = 'WrappedAction'

```
class dcs.task.EPLRS (group_id=1)
```

Bases: *dcs.task.WrappedAction*

**Key** = 'EPLRS'

**eplrs**

```
class dcs.task.ActivateBeaconCommand (channel=10, modechannel='X', callsign='TKR', bearing=True)
```

Bases: *dcs.task.WrappedAction*

**Key** = 'ActivateBeacon'

**static calc\_tacan\_frequency** (mode\_channel, channel, aa=True)

```
class dcs.task.RunScript (script: str='')
```

Bases: *dcs.task.WrappedAction*

Runs a given script string

**Parameters** **script** – to be executed

**Key** = 'Script'

```
class dcs.task.RunScriptFile (resourcekey: str='')
```

Bases: *dcs.task.WrappedAction*

Runs a script attached to the mission

**Parameters** **resourcekey** – resource key to the script file, see *dcs.mission.MapResource*

**Key** = 'ScriptFile'

```
class dcs.task.TransmitMessage (soundfile_reskey: typing.Union=None, subtitle_resstring: typing.Union=None, loop=False, subtitle_duration=5)
```

Bases: *dcs.task.WrappedAction*

Transmits a given sound file over the current radio channel.

**Parameters**

- **soundfile\_reskey** – resource key to the sound file to transmit, see *dcs.mission.MapResource*
- **subtitle\_resstring** – string resource key to subtitle displayed, see *dcs.mission.Mission.translation*
- **loop** – True or False if the sound file should be looped.
- **subtitle\_duration** – how long the subtitle should be displayed in seconds.

**Key = 'TransmitMessage'**

**class** *dcs.task.StopTransmission*

Bases: *dcs.task.WrappedAction*

Stops any *dcs.task.TransmitMessage* task currently ongoing.

**Key = 'StopTransmission'**

**class** *dcs.task.SetFrequencyCommand* (*frequency=133*, *modulation:*  
*dcs.task.Modulation=<Modulation.AM: 0>*)

Bases: *dcs.task.WrappedAction*

Set the groups radio frequency.

**Parameters**

- **frequency** – frequency band in mhz.
- **modulation** – AM or FM, see *dcs.task.Modulation*

**Key = 'SetFrequency'**

**class** *dcs.task.SwitchWaypoint* (*from\_waypoint=1*, *to\_waypoint=2*)

Bases: *dcs.task.WrappedAction*

Switch to a different waypoint.

**Parameters**

- **from\_waypoint** – from which waypoint to switch.??
- **to\_waypoint** – new current waypoint

**Key = 'SwitchWaypoint'**

**class** *dcs.task.SetInvisibleCommand* (*value=True*)

Bases: *dcs.task.WrappedAction*

**Key = 'SetInvisible'**

**class** *dcs.task.SetImmortalCommand* (*value=True*)

Bases: *dcs.task.WrappedAction*

**Key = 'SetImmortal'**

**class** *dcs.task.MainTask*

Bases: *object*

**name** = None

**sub\_tasks** = []

**perform\_task** = []

**map** = {'SEAD': <class 'dcs.task.SEAD'>, 'AFAC': <class 'dcs.task.AFAC'>, 'Nothing': <class 'dcs.task.Nothing'>, 'Refu

**class** *dcs.task.Nothing*

Bases: *dcs.task.MainTask*

**id** = 15



```
    name = 'Nothing'
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Aerobatics'>]
class dcs.task.AFAC
    Bases: dcs.task.MainTask
    id = 16
    name = 'AFAC'
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.AttackGroup'>, <class 'dcs.task.Aerobatics'>]
class dcs.task.AWACS
    Bases: dcs.task.MainTask
    id = 14
    name = 'AWACS'
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.RefuelingTaskAction'>]
    perform_task = [<class 'dcs.task.AWACSTaskAction'>]
class dcs.task.AntishipStrike
    Bases: dcs.task.MainTask
    id = 30
    name = 'Antiship Strike'
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.AttackGroup'>, <class 'dcs.task.Aerobatics'>]
    perform_task = [<class 'dcs.task.AntishipStrikeTaskAction'>]
class dcs.task.CAS
    Bases: dcs.task.MainTask
    id = 31
    name = 'CAS'
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.AttackGroup'>, <class 'dcs.task.Aerobatics'>]
    perform_task = [<class 'dcs.task.CASTaskAction'>]
class EnrouteTasks
    Bases: object
    class EngageGroup (group_id=0, visible=False)
        Bases: dcs.task.Task
        Id = 'EngageGroup'
    class CAS.EnrouteTasks.EngageTargetsInZone (position:
                                                dcs.mapping.Point=Point(0,
                                                0), radius=5000, targets: typing.List=None)
        Bases: dcs.task.Task
        Id = 'EngageTargetsInZone'
    class CAS.EnrouteTasks.EngageUnit (unit_id=0, visible=False)
        Bases: dcs.task.Task
        Id = 'EngageUnit'
```

```
class dcs.task.CAP
    Bases: dcs.task.MainTask

    id = 11

    name = 'CAP'

    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Aerobatics'>]

    perform_task = [<class 'dcs.task.CAPTaskAction'>]

    class EnrouteTasks
        Bases: object

        class EngageTargets (max_distance=None, targets: typing.List=None)
            Bases: dcs.task.Task

            Id = 'EngageTargets'

        class CAP.EnrouteTasks.EngageTargetsInZone (position: dcs.mapping.Point=Point(0, 0), radius=5000, targets: typing.List=None)
            Bases: dcs.task.Task

            Id = 'EngageTargetsInZone'

        class CAP.EnrouteTasks.EngageGroup (group_id=0, visible=False)
            Bases: dcs.task.Task

            Id = 'EngageGroup'

        class CAP.EnrouteTasks.EngageUnit (unit_id=0, visible=False)
            Bases: dcs.task.Task

            Id = 'EngageUnit'

class dcs.task.Escort
    Bases: dcs.task.MainTask

    id = 18

    name = 'Escort'

    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.EscortTaskAction'>]

    perform_task = [<class 'dcs.task.EscortTaskAction'>]

class dcs.task.FighterSweep
    Bases: dcs.task.MainTask

    id = 19

    name = 'Fighter Sweep'

    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Aerobatics'>]

    perform_task = [<class 'dcs.task.FighterSweepTaskAction'>]

class dcs.task.GroundAttack
    Bases: dcs.task.MainTask

    id = 32

    name = 'Ground Attack'

    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Bombing'>, <class 'dcs.task.Attack'>]
```

```
class dcs.task.Intercept
    Bases: dcs.task.MainTask
```

```
    id = 10
```

```
    name = 'Intercept'
```

```
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.AttackGroup'>, <class 'dcs.task.AttackGroup'>]
```

```
class dcs.task.PinpointStrike
    Bases: dcs.task.MainTask
```

```
    id = 33
```

```
    name = 'Pinpoint Strike'
```

```
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Bombing'>, <class 'dcs.task.AttackGroup'>]
```

```
class dcs.task.Reconnaissance
    Bases: dcs.task.MainTask
```

```
    id = 17
```

```
    name = 'Reconnaissance'
```

```
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Aerobatics'>]
```

```
    perform_task = []
```

```
class dcs.task.Refueling
    Bases: dcs.task.MainTask
```

```
    id = 13
```

```
    name = 'Refueling'
```

```
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>]
```

```
    perform_task = [<class 'dcs.task.Tanker'>]
```

```
class dcs.task.RunwayAttack
    Bases: dcs.task.MainTask
```

```
    id = 34
```

```
    name = 'Ground Attack'
```

```
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Bombing'>, <class 'dcs.task.AttackGroup'>]
```

```
    perform_task = []
```

```
class dcs.task.SEAD
    Bases: dcs.task.MainTask
```

```
    id = 29
```

```
    name = 'SEAD'
```

```
    sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.AttackGroup'>, <class 'dcs.task.AttackGroup'>]
```

```
    perform_task = [<class 'dcs.task.SEADTaskAction'>]
```

```
class dcs.task.Transport
    Bases: dcs.task.MainTask
```

```
    id = 35
```

```
    name = 'Transport'
```

```
sub_tasks = [<class 'dcs.task.OrbitAction'>, <class 'dcs.task.Follow'>, <class 'dcs.task.Aerobatics'>]
perform_task = []
```

**class** `dcs.task.Option` (*value=None*)  
Bases: `dcs.task.Task`

**Id** = 'WrappedAction'

**Key** = None

**class** `dcs.task.OptROE` (*value=0*)  
Bases: `dcs.task.Option`

Sets the groups rules of engagement. Ultimately defines whether or not the AI group is allowed to attack. This option can override all other tasking. With the 1.5 patch two of the values have different names in the mission editor. However the behavior is still exactly the same as before, its just labeled slightly different. The scripting engine still uses the previous values.

**Key** = 0

**class Values**  
Bases: `object`

**WeaponFree** = 0  
AI will engage any enemy group it detects. Target prioritization is based based on the threat of the target.

**OpenFireWeaponFree** = 1  
AI will engage any enemy group it detects, but will prioritize targets specified in the groups tasking.

**OpenFire** = 2  
AI will engage only targets specified in its taskings.

**ReturnFire** = 3  
AI will only engage threats that shoot first.

**WeaponHold** = 4  
AI will hold fire under all circumstances.

**class** `dcs.task.OptReactOnThreat` (*value=0*)  
Bases: `dcs.task.Option`

Defines the allowable action for an airborne group to take in response to a threat. This option can have an effect on other tasking.

**Key** = 1

**class Values**  
Bases: `object`

**NoReaction** = 0  
No defensive actions will take place to counter threats

**PassiveDefense** = 1  
AI will use jammers and other countermeasures in an attempt to defeat the threat. AI will not attempt a maneuver to defeat a threat.

**EvadeFire** = 2  
AI will react by performing defensive maneuvers against incoming threats, AI will also use passive defense.

**ByPassAndEscape = 3**

AI will attempt to avoid enemy threat zones all together. This includes attempting to fly above or around threats.

**AllowAbortMission = 4**

If a threat is deemed severe enough the AI will abort its mission and return to base.

**class** `dcs.task.OptDisperseUnderFire` (*value=None*)

Bases: `dcs.task.Option`

**Key = 8**



## 5.1 dcs package

### 5.1.1 Subpackages

#### dcs.lua package

##### Submodules

##### dcs.lua.parse module

`dcs.lua.parse.loads` (*tablestr*)

##### dcs.lua.serialize module

`dcs.lua.serialize.dumps` (*value, varname=None, indent=None*)

##### Module contents

#### dcs.terrain package

##### Submodules

##### dcs.terrain.caucasus module

```
class dcs.terrain.caucasus.Anapa
    Bases: dcs.terrain.terrain.Airport
    id = 12
    name = 'Anapa'
    position = Point(-5406.2803440839, 243127.2973737)
    tacan = None
    frequencies = [121000000, 38400000, 250000000, 3750000]
    unit_zones = []
```

```
    civilian = True

class dcs.terrain.caucasus.KrasnodarCenter
    Bases: dcs.terrain.terrain.Airport

    id = 13

    name = 'Krasnodar-Center'

    position = Point(11692.789495652, 367948.47230953)

    tacan = None

    frequencies = [122000000, 38600000, 251000000, 3800000]

    unit_zones = []

    civilian = True

class dcs.terrain.caucasus.Novorossiysk
    Bases: dcs.terrain.terrain.Airport

    id = 14

    name = 'Novorossiysk'

    position = Point(-40915.496728899, 279256.64920952)

    tacan = None

    frequencies = [123000000, 38800000, 252000000, 3850000]

    unit_zones = []

    civilian = True

class dcs.terrain.caucasus.Krymsk
    Bases: dcs.terrain.terrain.Airport

    id = 15

    name = 'Krymsk'

    position = Point(-6583.663574989, 294383.98405512)

    tacan = None

    frequencies = [124000000, 39000000, 253000000, 3900000]

    unit_zones = []

    civilian = True

class dcs.terrain.caucasus.Maykop
    Bases: dcs.terrain.terrain.Airport

    id = 16

    name = 'Maykop'

    position = Point(-26441.347360305, 458040.61422532)

    tacan = None

    frequencies = [125000000, 39200000, 254000000, 3950000]

    unit_zones = []

    civilian = True
```



```

class dcs.terrain.caucasus.Gelendzhik
    Bases: dcs.terrain.terrain.Airport

    id = 17

    name = 'Gelendzhik'

    position = Point(-50392.648146355, 298387.43849386)

    tacan = None

    frequencies = [126000000, 39400000, 255000000, 4000000]

    unit_zones = []

    civilian = True

class dcs.terrain.caucasus.Sochi
    Bases: dcs.terrain.terrain.Airport

    id = 18

    name = 'Sochi'

    position = Point(-164474.73482633, 462236.21834688)

    tacan = None

    frequencies = [127000000, 39600000, 256000000, 4050000]

    unit_zones = []

    civilian = True

class dcs.terrain.caucasus.KrasnodarPashkovsky
    Bases: dcs.terrain.terrain.Airport

    id = 19

    name = 'Krasnodar-Pashkovsky'

    position = Point(7674.038444859, 385029.5736699)

    tacan = None

    frequencies = [128000000, 39800000, 257000000, 4100000]

    unit_zones = []

    civilian = True

class dcs.terrain.caucasus.Sukhumi
    Bases: dcs.terrain.terrain.Airport

    id = 20

    name = 'Sukhumi'

    position = Point(-220531.73642658, 564387.05872916)

    tacan = None

    frequencies = [129000000, 40000000, 258000000, 4150000]

    unit_zones = []

    civilian = True

class dcs.terrain.caucasus.Gudauta
    Bases: dcs.terrain.terrain.Airport

```

```
    id = 21
    name = 'Gudauta'
    position = Point(-196974.19851241, 516290.23098695)
    tacan = None
    frequencies = [130000000, 40200000, 259000000, 4200000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Batumi
    Bases: dcs.terrain.terrain.Airport

    id = 22
    name = 'Batumi'
    position = Point(-355692.3067714, 617269.96285781)
    tacan = '16X'
    frequencies = [131000000, 40400000, 260000000, 4250000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Senaki
    Bases: dcs.terrain.terrain.Airport

    id = 23
    name = 'Senaki'
    position = Point(-281713.83114196, 647369.87369832)
    tacan = '31X'
    frequencies = [132000000, 40600000, 261000000, 4300000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Kobuleti
    Bases: dcs.terrain.terrain.Airport

    id = 24
    name = 'Kobuleti'
    position = Point(-317948.32727306, 635639.37385346)
    tacan = '67X'
    frequencies = [133000000, 40800000, 262000000, 4350000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Kutaisi
    Bases: dcs.terrain.terrain.Airport

    id = 25
```

```
name = 'Kutaisi'
position = Point(-284889.06283057, 683853.75717885)
tacan = None
frequencies = [134000000, 41000000, 263000000, 4400000]
unit_zones = []
civilian = True

class dcs.terrain.caucasus.Mineralnye
    Bases: dcs.terrain.terrain.Airport
    id = 26
    name = 'Mineralnye'
    position = Point(-51251.551717591, 705718.47981263)
    tacan = None
    frequencies = [135000000, 41200000, 264000000, 4450000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Nalchik
    Bases: dcs.terrain.terrain.Airport
    id = 27
    name = 'Nalchik'
    position = Point(-124921.90954665, 760428.0733062)
    tacan = None
    frequencies = [136000000, 41400000, 265000000, 4500000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Mozdok
    Bases: dcs.terrain.terrain.Airport
    id = 28
    name = 'Mozdok'
    position = Point(-83454.571428571, 834453.14285714)
    tacan = None
    frequencies = [137000000, 41600000, 266000000, 4550000]
    unit_zones = []
    civilian = False

class dcs.terrain.caucasus.Lochini
    Bases: dcs.terrain.terrain.Airport
    id = 29
    name = 'Lochini'
```

```
position = Point(-315478.57142857, 896538.85714286)
tacan = None
frequencies = [138000000, 41800000, 267000000, 4600000]
unit_zones = []
civilian = True

class dcs.terrain.caucasus.Soganlug
    Bases: dcs.terrain.terrain.Airport
    id = 30
    name = 'Soganlug'
    position = Point(-317838.57142857, 895424.57142858)
    tacan = None
    frequencies = [139000000, 42000000, 268000000, 4650000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Vaziani
    Bases: dcs.terrain.terrain.Airport
    id = 31
    name = 'Vaziani'
    position = Point(-319069.063, 903150.625)
    tacan = None
    frequencies = [140000000, 42200000, 269000000, 4700000]
    unit_zones = []
    civilian = False

class dcs.terrain.caucasus.Beslan
    Bases: dcs.terrain.terrain.Airport
    id = 32
    name = 'Beslan'
    position = Point(-148810.84954665, 843756.7533062)
    tacan = None
    frequencies = [141000000, 42400000, 270000000, 4750000]
    unit_zones = []
    civilian = True

class dcs.terrain.caucasus.Caucasus
    Bases: dcs.terrain.terrain.Terrain
    center = {'lat': 45.12945, 'long': 34.26527}
    bounds = Rectangle(380000, -560000, -600000, 1130000)
    map_view_default = <dcs.terrain.terrain.MapView object>
```

```

soganlug () → dcs.terrain.terrain.Airport
senaki () → dcs.terrain.terrain.Airport
sochi () → dcs.terrain.terrain.Airport
batumi () → dcs.terrain.terrain.Airport
nalchik () → dcs.terrain.terrain.Airport
beslan () → dcs.terrain.terrain.Airport
mozdok () → dcs.terrain.terrain.Airport
anapa () → dcs.terrain.terrain.Airport
krasnodar_center () → dcs.terrain.terrain.Airport
krasnodar_pashkovsky () → dcs.terrain.terrain.Airport
novorossiysk () → dcs.terrain.terrain.Airport
krymsk () → dcs.terrain.terrain.Airport
maykop () → dcs.terrain.terrain.Airport
gelendzhik () → dcs.terrain.terrain.Airport
mineralnye () → dcs.terrain.terrain.Airport
gudauta () → dcs.terrain.terrain.Airport
vaziani () → dcs.terrain.terrain.Airport
lochini () → dcs.terrain.terrain.Airport
kobuleti () → dcs.terrain.terrain.Airport
kutaisi () → dcs.terrain.terrain.Airport
sukhumi () → dcs.terrain.terrain.Airport
default_red_airports () → typing.List
default_blue_airports () → typing.List

```

#### **dcs.terrain.nevada module**

```

class dcs.terrain.nevada.Creech
    Bases: dcs.terrain.terrain.Airport
    id = 1
    name = 'Creech'
    position = Point(-359732, -74970.9)
    tacan = None
    frequencies = [122000000, 38600000, 251000000]
    unit_zones = []
    civilian = False
    slot_version = 2

```

```
class dcs.terrain.nevada.Groom
    Bases: dcs.terrain.terrain.Airport

    id = 2
    name = 'Groom'
    position = Point(-288694, -87414.2)
    tacan = None
    frequencies = [123000000, 38800000, 252000000]
    unit_zones = []
    civilian = False
    slot_version = 2

class dcs.terrain.nevada.McCarran
    Bases: dcs.terrain.terrain.Airport

    id = 3
    name = 'McCarran'
    position = Point(-416083, -27121.1)
    tacan = None
    frequencies = [124000000, 39000000, 253000000]
    unit_zones = []
    civilian = False
    slot_version = 2

class dcs.terrain.nevada.Nellis
    Bases: dcs.terrain.terrain.Airport

    id = 4
    name = 'Nellis'
    position = Point(-397971, -17639.5)
    tacan = None
    frequencies = [125000000, 39200000, 254000000]
    unit_zones = []
    civilian = False
    slot_version = 2

class dcs.terrain.nevada.Nevada
    Bases: dcs.terrain.terrain.Terrain

    center = {'lat': 39.81806, 'long': -114.73333}
    bounds = Rectangle(-497177.65625, -329334.875, -166934.953125, 209836.890625)
    map_view_default = <dcs.terrain.terrain.MapView object>
    creech () → dcs.terrain.terrain.Airport
    groom () → dcs.terrain.terrain.Airport
```

`mccarran()` → `dcs.terrain.terrain.Airport`

`nellis()` → `dcs.terrain.terrain.Airport`

### `dcs.terrain.terrain` module

**class** `dcs.terrain.terrain.ParkingSlot` (*crossroad\_idx, position: dcs.mapping.Point, large=False, slot\_name=None, heli=False, airplanes=True, length=None, width=None, height=None, shelter=False*)

Bases: `object`

**class** `dcs.terrain.terrain.Runway` (*heading, ils=None, leftright=0*)

Bases: `object`

**class** `dcs.terrain.terrain.Airport`

Bases: `object`

`id = None`

`name = None`

`position = None`

`tacan = None`

`frequencies = []`

`unit_zones = []`

`civilian = True`

`slot_version = 1`

`load_from_dict(d)`

`set_blue()`

`set_red()`

`set_neutral()`

`set_coalition(side)`

`is_red()`

`is_blue()`

`random_unit_zone()` → `dcs.mapping.Rectangle`

`free_parking_slots(aircraft_type: dcs.unittypes.FlyingType)` → `typing.List`

`free_parking_slot(aircraft_type: dcs.unittypes.FlyingType)` → `typing.Union`

`dict()`

**class** `dcs.terrain.terrain.MapView` (*center: dcs.mapping.Point, zoom=1000000*)

Bases: `object`

`load_from_dict(d)`

`dict()`

**class** `dcs.terrain.terrain.Terrain` (*name: str*)

Bases: `object`

`bounds = None`

```
map_view_default = None

airport_by_id (id: int) → dcs.terrain.terrain.Airport

airport_list () → typing.List

class dcs.terrain.terrain.Warehouses (terrain: dcs.terrain.terrain.Terrain)
    Bases: object

    load_dict (data)
```

## Module contents

### 5.1.2 Submodules

#### 5.1.3 dcs.country module

```
dcs.country.find_exact (group_name, find_name)
dcs.country.find_match (group_name, find_name)

class dcs.country.Country (_id, name)
    Bases: object

    callsign = {}
    planes = []
    helicopters = []
    add_vehicle_group (vgroup)
    add_ship_group (sgroup)
    add_plane_group (pgroup)
    add_helicopter_group (hgroup)
    add_aircraft_group (group: dcs.unitgroup.FlyingGroup)
    add_static_group (sgroup)
    find_group (group_name, search='exact')
    find_vehicle_group (name: str, search='exact')
    find_ship_group (name: str, search='exact')
    find_plane_group (name: str, search='exact')
    find_helicopter_group (name: str, search='exact')
    find_static_group (name: str, search='exact')
    next_callsign_id()
    next_callsign_category (category)
    dict ()
```



### 5.1.4 dcs.forcedoptions module

```

class dcs.forcedoptions.ForcedOptions
    Bases: object

    class Views
        Bases: enum.Enum

    class ForcedOptions.CivilTraffic
        Bases: enum.Enum

    class ForcedOptions.GEffect
        Bases: enum.Enum

    ForcedOptions.load_from_dict(d)

    ForcedOptions.dict()

```

### 5.1.5 dcs.goals module

```

class dcs.goals.GoalRule(predicate)
    Bases: object

    dict()

class dcs.goals.AllOfCoalitionInZone(coalitionlist, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_all_of_coalition_in_zone'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.AllOfCoalitionOutsideZone(coalitionlist, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_all_of_coalition_out_zone'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.AllOfGroupInZone(group, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_all_of_group_in_zone'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.AllOfGroupOutsideZone(group, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_all_of_group_out_zone'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.ArgumentInRange(argument, _min, _max)
    Bases: dcs.goals.GoalRule

    predicate = 'c_argument_in_range'

```

```
    classmethod create_from_dict (d)

    dict ()

class dcs.goals.BombInZone (typebomb, numbombs, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_bomb_in_zone'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.CargoUnhookedInZone (cargo, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_cargo_unhooked_in_zone'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.CoalitionHasAirdrome (coalitionlist, airdromelist)
    Bases: dcs.goals.GoalRule

    predicate = 'c_coalition_has_aidrome'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.CoalitionHasHelipad (coalitionlist, helipadlist)
    Bases: dcs.goals.GoalRule

    predicate = 'c_coalition_has_helipad'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.CockpitHighlightVisible (highlight_id)
    Bases: dcs.goals.GoalRule

    predicate = 'c_cockpit_highlight_visible'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.CockpitParamEqual (cockpit_param, value_text)
    Bases: dcs.goals.GoalRule

    predicate = 'c_cockpit_param_equal_to'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.CockpitParamEqualToAnother (cockpit_param, cockpit_param2)
    Bases: dcs.goals.GoalRule

    predicate = 'c_cockpit_param_is_equal_to_another'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.CockpitParamInRange (cockpit_param, _min2, _max2)
    Bases: dcs.goals.GoalRule
```

```

    predicate = 'c_cockpit_param_in_range'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.FlagEquals (flag=1, value=10)
    Bases: dcs.goals.GoalRule
    predicate = 'c_flag_equals'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.FlagEqualsFlag (flag=1, flag2=1)
    Bases: dcs.goals.GoalRule
    predicate = 'c_flag_equals_flag'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.FlagIsFalse (flag=1)
    Bases: dcs.goals.GoalRule
    predicate = 'c_flag_is_false'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.FlagIsLess (flag=1, value=10)
    Bases: dcs.goals.GoalRule
    predicate = 'c_flag_less'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.FlagIsLessThanFlag (flag=1, flag2=1)
    Bases: dcs.goals.GoalRule
    predicate = 'c_flag_less_flag'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.FlagIsMore (flag=1, value=10)
    Bases: dcs.goals.GoalRule
    predicate = 'c_flag_more'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.FlagIsTrue (flag=1)
    Bases: dcs.goals.GoalRule
    predicate = 'c_flag_is_true'
    classmethod create_from_dict (d)
    dict ()

```

```
class dcs.goals.GroupAlive(group)
    Bases: dcs.goals.GoalRule

    predicate = 'c_group_alive'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.GroupDead(group)
    Bases: dcs.goals.GoalRule

    predicate = 'c_group_dead'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.GroupLifeLess(group, percent=10)
    Bases: dcs.goals.GoalRule

    predicate = 'c_group_life_less'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.IndicationTextEqual(indicator_id, element_name, element_value)
    Bases: dcs.goals.GoalRule

    predicate = 'c_indication_txt_equal_to'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.MissionScoreHigher(coalitionlist, score=50)
    Bases: dcs.goals.GoalRule

    predicate = 'c_mission_score_higher'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.MissionScoreLower(coalitionlist, score=50)
    Bases: dcs.goals.GoalRule

    predicate = 'c_mission_score_lower'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.Or
    Bases: dcs.goals.GoalRule

    predicate = 'or'

    classmethod create_from_dict(d)

    dict()

class dcs.goals.PartOfCoalitionInZone(coalitionlist, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_part_of_coalition_in_zone'

    classmethod create_from_dict(d)
```

```

    dict ()

class dcs.goals.PartOfCoalitionOutsideZone (coalitionlist, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_part_of_coalition_out_zone'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.PartOfGroupInZone (group, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_part_of_group_in_zone'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.PartOfGroupOutsideZone (group, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_part_of_group_out_zone'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.PlayerScoreLess (scores=100)
    Bases: dcs.goals.GoalRule

    predicate = 'c_player_score_less'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.PlayerScoreMore (scores=100)
    Bases: dcs.goals.GoalRule

    predicate = 'c_player_score_more'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.Predicate (text)
    Bases: dcs.goals.GoalRule

    predicate = 'c_predicate'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.Random (percent=10)
    Bases: dcs.goals.GoalRule

    predicate = 'c_random_less'

    classmethod create_from_dict (d)

    dict ()

class dcs.goals.SignalFlareInZone (flare_color, numflares, zone)
    Bases: dcs.goals.GoalRule

    predicate = 'c_signal_flare_in_zone'

```

```
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.TimeAfter (seconds=10)
    Bases: dcs.goals.GoalRule
    predicate = 'c_time_after'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.TimeBefore (seconds=10)
    Bases: dcs.goals.GoalRule
    predicate = 'c_time_before'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.TimeSinceFlag (flag=1, seconds=10)
    Bases: dcs.goals.GoalRule
    predicate = 'c_time_since_flag'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitAlive (unit)
    Bases: dcs.goals.GoalRule
    predicate = 'c_unit_alive'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitAltitudeHigher (unit, altitude=1)
    Bases: dcs.goals.GoalRule
    predicate = 'c_unit_altitude_higher'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitAltitudeLower (unit, altitude=1)
    Bases: dcs.goals.GoalRule
    predicate = 'c_unit_altitude_lower'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitBankWithin (unit, min_unit_bank=-180, max_unit_bank=180)
    Bases: dcs.goals.GoalRule
    predicate = 'c_unit_bank'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitDamaged (unit)
    Bases: dcs.goals.GoalRule
```

```
    predicate = 'c_unit_damaged'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitDead (unit)
    Bases: dcsgoals.GoalRule
    predicate = 'c_unit_dead'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitHeadingWithin (unit, min_unit_heading, max_unit_heading=360)
    Bases: dcsgoals.GoalRule
    predicate = 'c_unit_heading'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitInMovingZone (unit, zone, zoneunit)
    Bases: dcsgoals.GoalRule
    predicate = 'c_unit_in_zone_unit'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitInZone (unit, zone)
    Bases: dcsgoals.GoalRule
    predicate = 'c_unit_in_zone'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitLifeLess (unit, percent=10)
    Bases: dcsgoals.GoalRule
    predicate = 'c_unit_life_less'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitOutsideMovingZone (unit, zone, zoneunit)
    Bases: dcsgoals.GoalRule
    predicate = 'c_unit_out_zone_unit'
    classmethod create_from_dict (d)
    dict ()

class dcs.goals.UnitOutsideZone (unit, zone)
    Bases: dcsgoals.GoalRule
    predicate = 'c_unit_out_zone'
    classmethod create_from_dict (d)
    dict ()
```

```
class dcs.goals.UnitPitchWithin (unit, min_unit_pitch=-180, max_unit_pitch=180)
```

```
    Bases: dcs.goals.GoalRule
```

```
    predicate = 'c_unit_pitch'
```

```
    classmethod create_from_dict (d)
```

```
    dict ()
```

```
class dcs.goals.UnitSpeedHigher (unit, speed=100)
```

```
    Bases: dcs.goals.GoalRule
```

```
    predicate = 'c_unit_speed_higher'
```

```
    classmethod create_from_dict (d)
```

```
    dict ()
```

```
class dcs.goals.UnitSpeedLower (unit, speed=100)
```

```
    Bases: dcs.goals.GoalRule
```

```
    predicate = 'c_unit_speed_lower'
```

```
    classmethod create_from_dict (d)
```

```
    dict ()
```

```
class dcs.goals.UnitVerticalSpeedWithin (unit, min_unit_vertical_speed=-300,
                                           max_unit_vertical_speed=300)
```

```
    Bases: dcs.goals.GoalRule
```

```
    predicate = 'c_unit_vertical_speed'
```

```
    classmethod create_from_dict (d)
```

```
    dict ()
```

```
class dcs.goals.Goal (comment='', score=100)
```

```
    Bases: object
```

```
    goalrule_map = {'c_unit_heading': <class 'dcs.goals.UnitHeadingWithin'>, 'c_coalition_has_helipad': <class 'dcs.goals.GoalHasHelipad'>}
```

```
    load_from_dict (data)
```

```
    conditions ()
```

```
    dict ()
```

```
class dcs.goals.Goals
```

```
    Bases: object
```

```
    load_from_dict (data)
```

```
    add_red (g: dcs.goals.Goal)
```

```
    add_blue (g: dcs.goals.Goal)
```

```
    add_offline (g: dcs.goals.Goal)
```

```
    generate_result ()
```

```
    dict ()
```



### 5.1.6 dcs.groundcontrol module

```
class dcs.groundcontrol.GroundControl
    Bases: object

    load_from_dict (d)

    dict ()
```

### 5.1.7 dcs.mapping module

```
dcs.mapping.point_from_heading (_x, _y, heading, distance)
    Calculates a point from a given point, heading and distance.
```

**Parameters**

- **\_x** – source point x
- **\_y** – source point y
- **heading** – heading in degrees from source point
- **distance** – distance from source point

**Returns** returns a tuple (x, y) of the calculated point

```
dcs.mapping.distance (x1, y1, x2, y2)
```

Returns the distance between 2 points

**Parameters**

- **x1** – x coordinate of point 1
- **y1** – y coordinate of point 1
- **x2** – x coordinate of point 2
- **y2** – y coordinate of point 2

**Returns** distance in point units(m)

```
dcs.mapping.heading_between_points (x1, y1, x2, y2)
```

Returns the angle between 2 points in degrees.

**Parameters**

- **x1** – x coordinate of point 1
- **y1** – y coordinate of point 1
- **x2** – x coordinate of point 2
- **y2** – y coordinate of point 2

**Returns** angle in degrees

```
class dcs.mapping.Point (x, y)
```

Bases: object

```
    point_from_heading (heading, distance)
```

```
    heading_between_point (point)
```

```
    distance_to_point (point)
```

```
class dcs.mapping.Triangle (points: typing.Union)
```

Bases: object

```
    area ()
    random_point () → dcs.mapping.Point
class dcs.mapping.Rectangle (top, left, bottom, right)
    Bases: object
    static from_point (point: dcs.mapping.Point, side_length)
    point_in_rect (point: dcs.mapping.Point)
    height ()
    width ()
    center () → dcs.mapping.Point
    resize (percentage: float)
    random_point () → dcs.mapping.Point
    random_distant_points (distance) → typing.Tuple
class dcs.mapping.Polygon (points: typing.List=None)
    Bases: object
    point_in_poly (point: dcs.mapping.Point)
        Checks if the given point is within the polygon.
        Parameters point – Point to test
        Returns True if point is within the polygon else False
    random_point () → dcs.mapping.Point
        Returns a random point within this polygon object
        Returns a random point
    static is_convex (a: dcs.mapping.Point, b: dcs.mapping.Point, c: dcs.mapping.Point)
    static in_triangle (a, b, c, p)
    is_clockwise ()
    static get_ear (poly)
    triangulate ()
    outbound_rectangle () → dcs.mapping.Rectangle
```

### 5.1.8 dcs.point module

```
class dcs.point.PointAction
    Bases: enum.Enum
class dcs.point.StaticPoint
    Bases: object
    load_from_dict (d, translation)
    dict ()
class dcs.point.VNav
    Bases: enum.Enum
class dcs.point.Scale
    Bases: enum.Enum
```

```

class dcs.point.Steer
    Bases: enum.Enum

class dcs.point.PointProperties (vnav:          dcs.point.VNav=<VNav.V2D: 0>,      scale:
                                     dcs.point.Scale=<Scale.Enroute: 0>,          steer:
                                     dcs.point.Steer=<Steer.ToTo: 2>, angle=None)

    Bases: object

    load_from_dict (d)

    dict ()

class dcs.point.MovingPoint
    Bases: dcs.point.StaticPoint

    load_from_dict (d, translation)

    find_task (task_type)
        Searches tasks in this point for the given task class

        Parameters task_type – task class to search, dcs.task

        Returns task instance if found, else None

    dict ()

```

### 5.1.9 dcs.templates module

```

class dcs.templates.VehicleTemplate
    Bases: object

    class Russia
        Bases: object

        static sa10_site (mission: dcs.mission.Mission, position: dcs.mapping.Point, heading, pre-
                           fix='', skill=<Skill.Average: 'Average'>)

    class VehicleTemplate.USA
        Bases: object

        static patriot_site (mission: dcs.mission.Mission, position, heading, prefix='',
                               skill=<Skill.Average: 'Average'>)

        static hawk_site (mission: dcs.mission.Mission, position, heading, prefix='',
                           skill=<Skill.Average: 'Average'>)

```

### 5.1.10 dcs.translation module

```

class dcs.translation.String (_id='', translation=None)
    Bases: object

    set (text, lang='DEFAULT')

    str (lang='DEFAULT')

class dcs.translation.Translation (_mission)
    Bases: object

    set_string (_id, string, lang='DEFAULT')

    get_string (_id)

    create_string (s, lang='DEFAULT')

```

```
delete_string(_id)
languages() → [<class 'str'>]
dict(lang='DEFAULT')
```

### 5.1.11 dcs.unit module

```
class dcs.unit.Skill
    Bases: enum.Enum

class dcs.unit.Unit(_id, name=None, type='')
    Bases: object

    load_from_dict(d)

    clone(_id)

    dict()

class dcs.unit.FlyingUnit(_id=None, name=None, _type: dcs.unittype.FlyingType=None, _country=None)
    Bases: dcs.unit.Unit

    load_from_dict(d)

    set_parking(parking_slot: dcs.terrain.terrain.ParkingSlot)

    load_pylon(weapon, pylon=None)

    store_loadout(filename)

    load_loadout(filename)

    reset_loadout()

    set_default_preset_channel(freq)

    set_radio_preset()

    set_player()

    set_client()

    dict()

class dcs.unit.Plane(_id=None, name=None, _type: dcs.planes.PlaneType=<class 'dcs.planes.A_10C'>, _country=None)
    Bases: dcs.unit.FlyingUnit

class dcs.unit.Helicopter(_id=None, name=None, _type: dcs.helicopters.HelicopterType=<class 'dcs.helicopters.Ka_50'>, _country=None)
    Bases: dcs.unit.FlyingUnit

    load_from_dict(d)

    dict()

class dcs.unit.Vehicle(id=None, name=None, _type='Sandbox')
    Bases: dcs.unit.Unit

    load_from_dict(d)

    dict()

class dcs.unit.Ship(id=None, name=None, _type=None)
    Bases: dcs.unit.Unit
```

```

    load_from_dict(d)

    dict()

class dcs.unit.Static(unit_id=None, name=None, _type: dcs.unittype.UnitType=None)
    Bases: dcs.unit.Unit

    load_from_dict(d)

    dict()

```

### 5.1.12 dcs.unitgroup module

```

class dcs.unitgroup.Group(_id: int, name=None)
    Bases: object

    class Formation
        Bases: enum.Enum

    Group.load_from_dict(d)

    Group.add_unit(unit: dcs.unit.Unit)

    Group.add_point(point: dcs.point.StaticPoint)

    Group.x

    Group.y

    Group.position

    Group.formation_line(heading, distance=20)

    Group.formation_star(heading, distance=20)

    Group.formation_rectangle(heading, distance=20)

    Group.formation_scattered(heading=0, max_radius=None)

    Group.formation(_type=<Formation.Line: 1>, heading=0)

    Group.set_skill(skill: dcs.unit.Skill)

    Group.dict()

class dcs.unitgroup.MovingGroup(_id, name=None, start_time=0)
    Bases: dcs.unitgroup.Group

    load_from_dict(d)

    dict()

class dcs.unitgroup.VehicleGroup(_id, name=None, start_time=0)
    Bases: dcs.unitgroup.MovingGroup

    load_from_dict(d)

    add_span(position: dcs.mapping.Point)

    add_waypoint(position: dcs.mapping.Point, move_formation: dcs.point.PointAction=<PointAction.OffRoad:
        'Off Road'>, speed=32) → dcs.point.MovingPoint

    dict()

class dcs.unitgroup.FlyingGroup(_id, name=None, start_time=0)
    Bases: dcs.unitgroup.MovingGroup

```

**starts\_from\_airport** () → bool

**load\_from\_dict** (d)

**add\_waypoint** (pos: *dcs.mapping.Point*, altitude, speed=600, name: *dcs.translation.String=None*)  
→ *dcs.point.MovingPoint*

**add\_runway\_waypoint** (airport: *dcs.terrain.terrain.Airport*, runway:  
*dcs.terrain.terrain.Runway=None*, distance=6400) →  
*dcs.point.MovingPoint*

Adds a waypoint parallel to the given runway heading, for start or approach.

#### Parameters

- **airport** – start airport object
- **runway** – runway for heading direction, if None first(default) airport runway will be used.
- **distance** – distance of the waypoint from the airport

**Returns** *MovePoint* object describing the waypoint

**land\_at** (airport: *dcs.terrain.terrain.Airport*) → *dcs.point.MovingPoint*

**load\_task\_default\_loadout** (task)

**load\_loadout** (name)

**load\_pylon** (weapon, pylon=None)

**load\_loadout\_from\_file** (filename)

**set\_client** ()

**reset\_loadout** ()

**set\_frequency** (frequency)

**dict** ()

**class** *dcs.unitgroup.PlaneGroup* (\_id, name=None, start\_time=0)

Bases: *dcs.unitgroup.FlyingGroup*

**add\_unit** (unit: *dcs.unit.Plane*)

**class** *dcs.unitgroup.HelicopterGroup* (\_id, name=None, start\_time=0)

Bases: *dcs.unitgroup.FlyingGroup*

**add\_unit** (unit: *dcs.unit.Helicopter*)

**class** *dcs.unitgroup.ShipGroup* (\_id, name=None, start\_time=0)

Bases: *dcs.unitgroup.MovingGroup*

**add\_waypoint** (position: *dcs.mapping.Point*, speed=20) → *dcs.point.MovingPoint*

**dict** ()

**class** *dcs.unitgroup.StaticGroup* (\_id, name=None)

Bases: *dcs.unitgroup.Group*

**load\_from\_dict** (d)

**dict** ()

### 5.1.13 dcs.unittype module

```
class dcs.unittype.UnitType
    Bases: object

    id = None

    name = None

class dcs.unittype.VehicleType
    Bases: dcs.unittype.UnitType

    eplrs = False

class dcs.unittype.ShipType
    Bases: dcs.unittype.UnitType

    helicopter_num = 0

    plane_num = 0

    parking = 0

class dcs.unittype.StaticType
    Bases: dcs.unittype.UnitType

    shape_name = None

    rate = 0

    category = 'Fortifications'

    sea_object = False

    can_cargo = False

    mass = None

class dcs.unittype.FlyingType
    Bases: dcs.unittype.UnitType

    flyable = False

    group_size_max = 4

    large_parking_slot = False

    helicopter = False

    fuel_max = 0

    max_speed = 500

    height = 0

    width = 0

    length = 0

    ammo_type = None

    chaff = 0

    flare = 0

    charge_total = 0

    chaff_charge_size = 1
```

```
flare_charge_size = 2
category = 'Air'
tacan = False
eplrs = False
radio_frequency = 251
panel_radio = None
pylons = {}
payloads = None
payload_dirs = ['C:\\Program Files\\Eagle Dynamics\\DCS World\\MissionEditor\\data\\scripts\\UnitPayloads', 'C:\\P
tasks = ['Nothing']
task_default = None
classmethod scan_payload_dir ()
classmethod load_payloads ()
classmethod loadout (_task)
classmethod loadout_by_name (loadout_name)
classmethod default_livery (country_name) → str
```

### 5.1.14 dcs.weather module

```
class dcs.weather.Wind (direction=0, speed=0)
    Bases: object
    dict ()

class dcs.weather.Cyclone
    Bases: object
    dict ()

class dcs.weather.Weather (terrain)
    Bases: object
    class Season
        Bases: object
        Summer = 1
        Winter = 2
        Spring = 3
        Fall = 4
    class Weather.BariSystem
        Bases: enum.Enum
    Weather.load_from_dict (d)
    Weather.set_season_from_datetime (dt: datetime.datetime)
    Weather.dynamic_weather (system: dcs.weather.BariSystem, cyclones: int=1)
    Weather.dict ()
```



### 5.1.15 Module contents



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## d

- `dcS`, 61
- `dcS.country`, 44
- `dcS.forcedoptions`, 45
- `dcS.goals`, 45
- `dcS.groundcontrol`, 53
- `dcS.lua`, 35
- `dcS.lua.parse`, 35
- `dcS.lua.serialize`, 35
- `dcS.mapping`, 53
- `dcS.mission`, 7
- `dcS.point`, 54
- `dcS.task`, 19
- `dcS.templates`, 55
- `dcS.terrain`, 44
- `dcS.terrain.caucasus`, 35
- `dcS.terrain.nevada`, 41
- `dcS.terrain.terrain`, 43
- `dcS.translation`, 55
- `dcS.unit`, 56
- `dcS.unitgroup`, 57
- `dcS.unittype`, 59
- `dcS.weather`, 60



## A

ActivateBeaconCommand (class in dcs.task), 27  
 add\_aircraft\_group() (dcs.country.Country method), 44  
 add\_blue() (dcs.goals.Goals method), 52  
 add\_helicopter\_group() (dcs.country.Country method), 44  
 add\_offline() (dcs.goals.Goals method), 52  
 add\_picture\_blue() (dcs.mission.Mission method), 8  
 add\_picture\_red() (dcs.mission.Mission method), 8  
 add\_plane\_group() (dcs.country.Country method), 44  
 add\_point() (dcs.unitgroup.Group method), 57  
 add\_red() (dcs.goals.Goals method), 52  
 add\_resource\_file() (dcs.mission.MapResource method), 17  
 add\_runway\_waypoint() (dcs.unitgroup.FlyingGroup method), 58  
 add\_ship\_group() (dcs.country.Country method), 44  
 add\_span() (dcs.unitgroup.VehicleGroup method), 57  
 add\_static\_group() (dcs.country.Country method), 44  
 add\_unit() (dcs.unitgroup.Group method), 57  
 add\_unit() (dcs.unitgroup.HelicopterGroup method), 58  
 add\_unit() (dcs.unitgroup.PlaneGroup method), 58  
 add\_vehicle\_group() (dcs.country.Country method), 44  
 add\_waypoint() (dcs.unitgroup.FlyingGroup method), 58  
 add\_waypoint() (dcs.unitgroup.ShipGroup method), 58  
 add\_waypoint() (dcs.unitgroup.VehicleGroup method), 57  
 Aerobatics (class in dcs.task), 25  
 AFAC (class in dcs.task), 29  
 aircraft() (dcs.mission.Mission method), 12  
 Airport (class in dcs.terrain.terrain), 43  
 airport\_by\_id() (dcs.terrain.terrain.Terrain method), 44  
 airport\_list() (dcs.terrain.terrain.Terrain method), 44  
 AllofCoalitionInZone (class in dcs.goals), 45  
 AllofCoalitionOutsideZone (class in dcs.goals), 45  
 AllofGroupInZone (class in dcs.goals), 45  
 AllofGroupOutsideZone (class in dcs.goals), 45  
 AllowAbortMission (dcs.task.OptReactOnThreat.Values attribute), 33  
 AM (dcs.task.Modulation attribute), 19

ammo\_type (dcs.unittype.FlyingType attribute), 59  
 Anapa (class in dcs.terrain.caucasus), 35  
 anapa() (dcs.terrain.caucasus.Caucasus method), 41  
 AntishipStrike (class in dcs.task), 29  
 AntishipStrikeTaskAction (class in dcs.task), 23  
 area() (dcs.mapping.Triangle method), 53  
 ArgumentInRange (class in dcs.goals), 45  
 AttackGroup (class in dcs.task), 23  
 AttackMapObject (class in dcs.task), 23  
 AttackUnit (class in dcs.task), 23  
 AWACS (class in dcs.task), 29  
 awacs\_flight() (dcs.mission.Mission method), 14  
 AWACSTaskAction (class in dcs.task), 25

## B

Batumi (class in dcs.terrain.caucasus), 38  
 batumi() (dcs.terrain.caucasus.Caucasus method), 41  
 Beslan (class in dcs.terrain.caucasus), 40  
 beslan() (dcs.terrain.caucasus.Caucasus method), 41  
 Bombing (class in dcs.task), 24  
 BombingRunway (class in dcs.task), 24  
 BombInZone (class in dcs.goals), 46  
 bounds (dcs.terrain.caucasus.Caucasus attribute), 40  
 bounds (dcs.terrain.nevada.Nevada attribute), 42  
 bounds (dcs.terrain.terrain.Terrain attribute), 43  
 ByPassAndEscape (dcs.task.OptReactOnThreat.Values attribute), 32

## C

calc\_tacan\_frequency() (dcs.task.ActivateBeaconCommand static method), 27  
 callsign (dcs.country.Country attribute), 44  
 can\_cargo (dcs.unittype.StaticType attribute), 59  
 CAP (class in dcs.task), 29  
 CAP.EnrouteTasks (class in dcs.task), 30  
 CAP.EnrouteTasks.EngageGroup (class in dcs.task), 30  
 CAP.EnrouteTasks.EngageTargets (class in dcs.task), 30  
 CAP.EnrouteTasks.EngageTargetsInZone (class in dcs.task), 30  
 CAP.EnrouteTasks.EngageUnit (class in dcs.task), 30

CAPTaskAction (class in dcs.task), 24  
CargoTransportation (class in dcs.task), 27  
CargoUnhookedInZone (class in dcs.goals), 46  
CAS (class in dcs.task), 29  
CAS.EnrouteTasks (class in dcs.task), 29  
CAS.EnrouteTasks.EngageGroup (class in dcs.task), 29  
CAS.EnrouteTasks.EngageTargetsInZone (class in dcs.task), 29  
CAS.EnrouteTasks.EngageUnit (class in dcs.task), 29  
CASTaskAction (class in dcs.task), 24  
category (dcs.unittype.FlyingType attribute), 60  
category (dcs.unittype.StaticType attribute), 59  
Caucasus (class in dcs.terrain.caucasus), 40  
center (dcs.terrain.caucasus.Caucasus attribute), 40  
center (dcs.terrain.nevada.Nevada attribute), 42  
center() (dcs.mapping.Rectangle method), 54  
chaff (dcs.unittype.FlyingType attribute), 59  
chaff\_charge\_size (dcs.unittype.FlyingType attribute), 59  
charge\_total (dcs.unittype.FlyingType attribute), 59  
civilian (dcs.terrain.caucasus.Anapa attribute), 35  
civilian (dcs.terrain.caucasus.Batumi attribute), 38  
civilian (dcs.terrain.caucasus.Beslan attribute), 40  
civilian (dcs.terrain.caucasus.Gelendzhik attribute), 37  
civilian (dcs.terrain.caucasus.Gudauta attribute), 38  
civilian (dcs.terrain.caucasus.Kobuleti attribute), 38  
civilian (dcs.terrain.caucasus.KrasnodarCenter attribute), 36  
civilian (dcs.terrain.caucasus.KrasnodarPashkovsky attribute), 37  
civilian (dcs.terrain.caucasus.Krymsk attribute), 36  
civilian (dcs.terrain.caucasus.Kutaisi attribute), 39  
civilian (dcs.terrain.caucasus.Lochini attribute), 40  
civilian (dcs.terrain.caucasus.Maykop attribute), 36  
civilian (dcs.terrain.caucasus.Mineralnye attribute), 39  
civilian (dcs.terrain.caucasus.Mozdok attribute), 39  
civilian (dcs.terrain.caucasus.Nalchik attribute), 39  
civilian (dcs.terrain.caucasus.Novorossiysk attribute), 36  
civilian (dcs.terrain.caucasus.Senaki attribute), 38  
civilian (dcs.terrain.caucasus.Sochi attribute), 37  
civilian (dcs.terrain.caucasus.Soganlug attribute), 40  
civilian (dcs.terrain.caucasus.Sukhumi attribute), 37  
civilian (dcs.terrain.caucasus.Vaziani attribute), 40  
civilian (dcs.terrain.nevada.Creech attribute), 41  
civilian (dcs.terrain.nevada.Groom attribute), 42  
civilian (dcs.terrain.nevada.McCarran attribute), 42  
civilian (dcs.terrain.nevada.Nellis attribute), 42  
civilian (dcs.terrain.terrain.Airport attribute), 43  
clone() (dcs.unit.Unit method), 56  
CoalitionHasAirdrome (class in dcs.goals), 46  
CoalitionHasHelipad (class in dcs.goals), 46  
CockpitHighlightVisible (class in dcs.goals), 46  
CockpitParamEqual (class in dcs.goals), 46  
CockpitParamEqualToAnother (class in dcs.goals), 46  
CockpitParamInRange (class in dcs.goals), 46  
Cold (dcs.mission.StartType attribute), 7  
conditions() (dcs.goals.Goal method), 52  
ControlledTask (class in dcs.task), 19  
Country (class in dcs.country), 44  
country() (dcs.mission.Mission method), 16  
create\_from\_dict() (dcs.goals.AllOfCoalitionInZone class method), 45  
create\_from\_dict() (dcs.goals.AllOfCoalitionOutsideZone class method), 45  
create\_from\_dict() (dcs.goals.AllOfGroupInZone class method), 45  
create\_from\_dict() (dcs.goals.AllOfGroupOutsideZone class method), 45  
create\_from\_dict() (dcs.goals.ArgumentInRange class method), 45  
create\_from\_dict() (dcs.goals.BombInZone class method), 46  
create\_from\_dict() (dcs.goals.CargoUnhookedInZone class method), 46  
create\_from\_dict() (dcs.goals.CoalitionHasAirdrome class method), 46  
create\_from\_dict() (dcs.goals.CoalitionHasHelipad class method), 46  
create\_from\_dict() (dcs.goals.CockpitHighlightVisible class method), 46  
create\_from\_dict() (dcs.goals.CockpitParamEqual class method), 46  
create\_from\_dict() (dcs.goals.CockpitParamEqualToAnother class method), 46  
create\_from\_dict() (dcs.goals.CockpitParamInRange class method), 47  
create\_from\_dict() (dcs.goals.FlagEquals class method), 47  
create\_from\_dict() (dcs.goals.FlagEqualsFlag class method), 47  
create\_from\_dict() (dcs.goals.FlagIsFalse class method), 47  
create\_from\_dict() (dcs.goals.FlagIsLess class method), 47  
create\_from\_dict() (dcs.goals.FlagIsLessThanFlag class method), 47  
create\_from\_dict() (dcs.goals.FlagIsMore class method), 47  
create\_from\_dict() (dcs.goals.FlagIsTrue class method), 47  
create\_from\_dict() (dcs.goals.GroupAlive class method), 48  
create\_from\_dict() (dcs.goals.GroupDead class method), 48  
create\_from\_dict() (dcs.goals.GroupLifeLess class method), 48  
create\_from\_dict() (dcs.goals.IndicationTextEqual class method), 48



- [create\\_from\\_dict\(\) \(dcs.goals.MissionScoreHigher class method\), 48](#)  
[create\\_from\\_dict\(\) \(dcs.goals.MissionScoreLower class method\), 48](#)  
[create\\_from\\_dict\(\) \(dcs.goals.Or class method\), 48](#)  
[create\\_from\\_dict\(\) \(dcs.goals.PartOfCoalitionInZone class method\), 48](#)  
[create\\_from\\_dict\(\) \(dcs.goals.PartOfCoalitionOutsideZone class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.PartOfGroupInZone class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.PartOfGroupOutsideZone class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.PlayerScoreLess class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.PlayerScoreMore class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.Predicate class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.Random class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.SignalFlareInZone class method\), 49](#)  
[create\\_from\\_dict\(\) \(dcs.goals.TimeAfter class method\), 50](#)  
[create\\_from\\_dict\(\) \(dcs.goals.TimeBefore class method\), 50](#)  
[create\\_from\\_dict\(\) \(dcs.goals.TimeSinceFlag class method\), 50](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitAlive class method\), 50](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitAltitudeHigher class method\), 50](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitAltitudeLower class method\), 50](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitBankWithin class method\), 50](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitDamaged class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitDead class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitHeadingWithin class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitInMovingZone class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitInZone class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitLifeLess class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitOutsideMovingZone class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitOutsideZone class method\), 51](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitPitchWithin class method\), 52](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitSpeedHigher class method\), 52](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitSpeedLower class method\), 52](#)  
[create\\_from\\_dict\(\) \(dcs.goals.UnitVerticalSpeedWithin class method\), 52](#)  
[create\\_from\\_dict\(\) \(dcs.task.Task class method\), 19](#)  
[create\\_string\(\) \(dcs.translation.Translation method\), 55](#)  
[Creech \(class in dcs.terrain.nevada\), 41](#)  
[creech\(\) \(dcs.terrain.nevada.Nevada method\), 42](#)  
[Cyclone \(class in dcs.weather\), 60](#)
- ## D
- [dcs \(module\), 61](#)  
[dcs.country \(module\), 44](#)  
[dcs.forcedoptions \(module\), 45](#)  
[dcs.goals \(module\), 45](#)  
[dcs.groundcontrol \(module\), 53](#)  
[dcs.lua \(module\), 35](#)  
[dcs.lua.parse \(module\), 35](#)  
[dcs.lua.serialize \(module\), 35](#)  
[dcs.mapping \(module\), 53](#)  
[dcs.mission \(module\), 7](#)  
[dcs.point \(module\), 54](#)  
[dcs.task \(module\), 19](#)  
[dcs.templates \(module\), 55](#)  
[dcs.terrain \(module\), 44](#)  
[dcs.terrain.caucasus \(module\), 35](#)  
[dcs.terrain.nevada \(module\), 41](#)  
[dcs.terrain.terrain \(module\), 43](#)  
[dcs.translation \(module\), 55](#)  
[dcs.unit \(module\), 56](#)  
[dcs.unitgroup \(module\), 57](#)  
[dcs.unittype \(module\), 59](#)  
[dcs.weather \(module\), 60](#)  
[default\\_blue\\_airports\(\) \(dcs.terrain.caucasus.Caucasus method\), 41](#)  
[default\\_livery\(\) \(dcs.unittype.FlyingType class method\), 60](#)  
[default\\_red\\_airports\(\) \(dcs.terrain.caucasus.Caucasus method\), 41](#)  
[delete\\_string\(\) \(dcs.translation.Translation method\), 55](#)  
[description\\_bluetask\\_text\(\) \(dcs.mission.Mission method\), 8](#)  
[description\\_redtask\\_text\(\) \(dcs.mission.Mission method\), 8](#)  
[description\\_text\(\) \(dcs.mission.Mission method\), 8](#)  
[Designation \(class in dcs.task\), 25](#)  
[dict\(\) \(dcs.country.Country method\), 44](#)  
[dict\(\) \(dcs.forcedoptions.ForcedOptions method\), 45](#)  
[dict\(\) \(dcs.goals.AllOfCoalitionInZone method\), 45](#)  
[dict\(\) \(dcs.goals.AllOfCoalitionOutsideZone method\), 45](#)  
[dict\(\) \(dcs.goals.AllOfGroupInZone method\), 45](#)  
[dict\(\) \(dcs.goals.AllOfGroupOutsideZone method\), 45](#)  
[dict\(\) \(dcs.goals.ArgumentInRange method\), 46](#)  
[dict\(\) \(dcs.goals.BombInZone method\), 46](#)

`dict()` (`dcs.goals.CargoUnhookedInZone` method), 46  
`dict()` (`dcs.goals.CoalitionHasAirdrome` method), 46  
`dict()` (`dcs.goals.CoalitionHasHelipad` method), 46  
`dict()` (`dcs.goals.CockpitHighlightVisible` method), 46  
`dict()` (`dcs.goals.CockpitParamEqual` method), 46  
`dict()` (`dcs.goals.CockpitParamEqualToAnother` method), 46  
`dict()` (`dcs.goals.CockpitParamInRange` method), 47  
`dict()` (`dcs.goals.FlagEquals` method), 47  
`dict()` (`dcs.goals.FlagEqualsFlag` method), 47  
`dict()` (`dcs.goals.FlagIsFalse` method), 47  
`dict()` (`dcs.goals.FlagIsLess` method), 47  
`dict()` (`dcs.goals.FlagIsLessThanFlag` method), 47  
`dict()` (`dcs.goals.FlagIsMore` method), 47  
`dict()` (`dcs.goals.FlagIsTrue` method), 47  
`dict()` (`dcs.goals.Goal` method), 52  
`dict()` (`dcs.goals.GoalRule` method), 45  
`dict()` (`dcs.goals.Goals` method), 52  
`dict()` (`dcs.goals.GroupAlive` method), 48  
`dict()` (`dcs.goals.GroupDead` method), 48  
`dict()` (`dcs.goals.GroupLifeLess` method), 48  
`dict()` (`dcs.goals.IndicationTextEqual` method), 48  
`dict()` (`dcs.goals.MissionScoreHigher` method), 48  
`dict()` (`dcs.goals.MissionScoreLower` method), 48  
`dict()` (`dcs.goals.Or` method), 48  
`dict()` (`dcs.goals.PartOfCoalitionInZone` method), 48  
`dict()` (`dcs.goals.PartOfCoalitionOutsideZone` method), 49  
`dict()` (`dcs.goals.PartOfGroupInZone` method), 49  
`dict()` (`dcs.goals.PartOfGroupOutsideZone` method), 49  
`dict()` (`dcs.goals.PlayerScoreLess` method), 49  
`dict()` (`dcs.goals.PlayerScoreMore` method), 49  
`dict()` (`dcs.goals.Predicate` method), 49  
`dict()` (`dcs.goals.Random` method), 49  
`dict()` (`dcs.goals.SignalFlareInZone` method), 50  
`dict()` (`dcs.goals.TimeAfter` method), 50  
`dict()` (`dcs.goals.TimeBefore` method), 50  
`dict()` (`dcs.goals.TimeSinceFlag` method), 50  
`dict()` (`dcs.goals.UnitAlive` method), 50  
`dict()` (`dcs.goals.UnitAltitudeHigher` method), 50  
`dict()` (`dcs.goals.UnitAltitudeLower` method), 50  
`dict()` (`dcs.goals.UnitBankWithin` method), 50  
`dict()` (`dcs.goals.UnitDamaged` method), 51  
`dict()` (`dcs.goals.UnitDead` method), 51  
`dict()` (`dcs.goals.UnitHeadingWithin` method), 51  
`dict()` (`dcs.goals.UnitInMovingZone` method), 51  
`dict()` (`dcs.goals.UnitInZone` method), 51  
`dict()` (`dcs.goals.UnitLifeLess` method), 51  
`dict()` (`dcs.goals.UnitOutsideMovingZone` method), 51  
`dict()` (`dcs.goals.UnitOutsideZone` method), 51  
`dict()` (`dcs.goals.UnitPitchWithin` method), 52  
`dict()` (`dcs.goals.UnitSpeedHigher` method), 52  
`dict()` (`dcs.goals.UnitSpeedLower` method), 52  
`dict()` (`dcs.goals.UnitVerticalSpeedWithin` method), 52  
`dict()` (`dcs.groundcontrol.GroundControl` method), 53  
`dict()` (`dcs.mission.Mission` method), 17  
`dict()` (`dcs.point.MovingPoint` method), 55  
`dict()` (`dcs.point.PointProperties` method), 55  
`dict()` (`dcs.point.StaticPoint` method), 54  
`dict()` (`dcs.task.AntishipStrikeTaskAction` method), 24  
`dict()` (`dcs.task.CAPTTaskAction` method), 24  
`dict()` (`dcs.task.CASTaskAction` method), 24  
`dict()` (`dcs.task.FighterSweepTaskAction` method), 24  
`dict()` (`dcs.task.SEADTaskAction` method), 24  
`dict()` (`dcs.task.Task` method), 19  
`dict()` (`dcs.terrain.terrain.Airport` method), 43  
`dict()` (`dcs.terrain.terrain.MapView` method), 43  
`dict()` (`dcs.translation.Translation` method), 56  
`dict()` (`dcs.unit.FlyingUnit` method), 56  
`dict()` (`dcs.unit.Helicopter` method), 56  
`dict()` (`dcs.unit.Ship` method), 57  
`dict()` (`dcs.unit.Static` method), 57  
`dict()` (`dcs.unit.Unit` method), 56  
`dict()` (`dcs.unit.Vehicle` method), 56  
`dict()` (`dcs.unitgroup.FlyingGroup` method), 58  
`dict()` (`dcs.unitgroup.Group` method), 57  
`dict()` (`dcs.unitgroup.MovingGroup` method), 57  
`dict()` (`dcs.unitgroup.ShipGroup` method), 58  
`dict()` (`dcs.unitgroup.StaticGroup` method), 58  
`dict()` (`dcs.unitgroup.VehicleGroup` method), 57  
`dict()` (`dcs.weather.Cyclone` method), 60  
`dict()` (`dcs.weather.Weather` method), 60  
`dict()` (`dcs.weather.Wind` method), 60  
`DisembarkFromTransport` (class in `dcs.task`), 26  
`distance()` (in module `dcs.mapping`), 53  
`distance_to_point()` (`dcs.mapping.Point` method), 53  
`dumps()` (in module `dcs.lua.serialize`), 35  
`dynamic_weather()` (`dcs.weather.Weather` method), 60

## E

`Embarking` (class in `dcs.task`), 26  
`EmbarkToTransport` (class in `dcs.task`), 26  
`EngageGroup` (class in `dcs.task`), 25  
`EngageTargets` (class in `dcs.task`), 24  
`EngageTargetsInZone` (class in `dcs.task`), 25  
`EngageUnit` (class in `dcs.task`), 25  
`EPLRS` (class in `dcs.task`), 27  
`eplrs` (`dcs.task.EPLRS` attribute), 27  
`eplrs` (`dcs.unittypes.FlyingType` attribute), 60  
`eplrs` (`dcs.unittypes.VehicleType` attribute), 59  
`eplrs_for()` (`dcs.mission.Mission` method), 9  
`Escort` (class in `dcs.task`), 30  
`escort_flight()` (`dcs.mission.Mission` method), 15  
`EscortTaskAction` (class in `dcs.task`), 24  
`EvadeFire` (`dcs.task.OptReactOnThreat.Values` attribute), 32  
`Expend` (class in `dcs.task`), 24

## F

FAC (class in dcs.task), 26  
 FACEngageGroup (class in dcs.task), 26  
 Fall (dcs.weather.Weather.Season attribute), 60  
 FighterSweep (class in dcs.task), 30  
 FighterSweepTaskAction (class in dcs.task), 24  
 find\_exact() (in module dcs.country), 44  
 find\_group() (dcs.country.Country method), 44  
 find\_group() (dcs.mission.Mission method), 16  
 find\_helicopter\_group() (dcs.country.Country method), 44  
 find\_match() (in module dcs.country), 44  
 find\_plane\_group() (dcs.country.Country method), 44  
 find\_ship\_group() (dcs.country.Country method), 44  
 find\_static\_group() (dcs.country.Country method), 44  
 find\_task() (dcs.point.MovingPoint method), 55  
 find\_vehicle\_group() (dcs.country.Country method), 44  
 FireAtPoint (class in dcs.task), 25  
 FlagEquals (class in dcs.goals), 47  
 FlagEqualsFlag (class in dcs.goals), 47  
 FlagIsFalse (class in dcs.goals), 47  
 FlagIsLess (class in dcs.goals), 47  
 FlagIsLessThanFlag (class in dcs.goals), 47  
 FlagIsMore (class in dcs.goals), 47  
 FlagIsTrue (class in dcs.goals), 47  
 flare (dcs.unittypes.FlyingType attribute), 59  
 flare\_charge\_size (dcs.unittypes.FlyingType attribute), 59  
 flight\_group() (dcs.mission.Mission method), 14  
 flight\_group\_from\_airport() (dcs.mission.Mission method), 13  
 flight\_group\_from\_unit() (dcs.mission.Mission method), 13  
 flight\_group\_inflight() (dcs.mission.Mission method), 12  
 flyable (dcs.unittypes.FlyingType attribute), 59  
 FlyingGroup (class in dcs.unitgroup), 57  
 FlyingType (class in dcs.unittypes), 59  
 FlyingUnit (class in dcs.unit), 56  
 FM (dcs.task.Modulation attribute), 19  
 Follow (class in dcs.task), 25  
 ForcedOptions (class in dcs.forcedoptions), 45  
 ForcedOptions.CivilTraffic (class in dcs.forcedoptions), 45  
 ForcedOptions.GEffect (class in dcs.forcedoptions), 45  
 ForcedOptions.Views (class in dcs.forcedoptions), 45  
 formation() (dcs.unitgroup.Group method), 57  
 formation\_line() (dcs.unitgroup.Group method), 57  
 formation\_rectangle() (dcs.unitgroup.Group method), 57  
 formation\_scattered() (dcs.unitgroup.Group method), 57  
 formation\_star() (dcs.unitgroup.Group method), 57  
 free\_parking\_slot() (dcs.terrain.terrain.Airport method), 43  
 free\_parking\_slots() (dcs.terrain.terrain.Airport method), 43  
 frequencies (dcs.terrain.caucasus.Anapa attribute), 35

frequencies (dcs.terrain.caucasus.Batumi attribute), 38  
 frequencies (dcs.terrain.caucasus.Beslan attribute), 40  
 frequencies (dcs.terrain.caucasus.Gelendzhik attribute), 37  
 frequencies (dcs.terrain.caucasus.Gudauta attribute), 38  
 frequencies (dcs.terrain.caucasus.Kobuleti attribute), 38  
 frequencies (dcs.terrain.caucasus.KrasnodarCenter attribute), 36  
 frequencies (dcs.terrain.caucasus.KrasnodarPashkovsky attribute), 37  
 frequencies (dcs.terrain.caucasus.Krymsk attribute), 36  
 frequencies (dcs.terrain.caucasus.Kutaisi attribute), 39  
 frequencies (dcs.terrain.caucasus.Lochini attribute), 40  
 frequencies (dcs.terrain.caucasus.Maykop attribute), 36  
 frequencies (dcs.terrain.caucasus.Mineralnye attribute), 39  
 frequencies (dcs.terrain.caucasus.Mozdok attribute), 39  
 frequencies (dcs.terrain.caucasus.Nalchik attribute), 39  
 frequencies (dcs.terrain.caucasus.Novorossiysk attribute), 36  
 frequencies (dcs.terrain.caucasus.Senaki attribute), 38  
 frequencies (dcs.terrain.caucasus.Sochi attribute), 37  
 frequencies (dcs.terrain.caucasus.Soganlug attribute), 40  
 frequencies (dcs.terrain.caucasus.Sukhumi attribute), 37  
 frequencies (dcs.terrain.caucasus.Vaziani attribute), 40  
 frequencies (dcs.terrain.nevada.Creech attribute), 41  
 frequencies (dcs.terrain.nevada.Groom attribute), 42  
 frequencies (dcs.terrain.nevada.McCarran attribute), 42  
 frequencies (dcs.terrain.nevada.Nellis attribute), 42  
 frequencies (dcs.terrain.terrain.Airport attribute), 43  
 from\_point() (dcs.mapping.Rectangle static method), 54  
 from\_string() (dcs.mission.StartType static method), 7  
 fuel\_max (dcs.unittypes.FlyingType attribute), 59

## G

Gelendzhik (class in dcs.terrain.caucasus), 36  
 gelendzhik() (dcs.terrain.caucasus.Caucasus method), 41  
 generate\_result() (dcs.goals.Goals method), 52  
 get\_ear() (dcs.mapping.Polygon static method), 54  
 get\_string() (dcs.translation.Translation method), 55  
 Goal (class in dcs.goals), 52  
 GoalRule (class in dcs.goals), 45  
 goalrule\_map (dcs.goals.Goal attribute), 52  
 Goals (class in dcs.goals), 52  
 Groom (class in dcs.terrain.nevada), 41  
 groom() (dcs.terrain.nevada.Nevada method), 42  
 GroundAttack (class in dcs.task), 30  
 GroundControl (class in dcs.groundcontrol), 53  
 Group (class in dcs.unitgroup), 57  
 Group.Formation (class in dcs.unitgroup), 57  
 group\_size\_max (dcs.unittypes.FlyingType attribute), 59  
 GroupAlive (class in dcs.goals), 47  
 GroupDead (class in dcs.goals), 48  
 GroupLifeLess (class in dcs.goals), 48

Gudauta (class in dcs.terrain.caucasus), 37  
gudauta() (dcs.terrain.caucasus.Caucasus method), 41

## H

hawk\_site() (dcs.templates.VehicleTemplate.USA static method), 55  
heading\_between\_point() (dcs.mapping.Point method), 53  
heading\_between\_points() (in module dcs.mapping), 53  
height (dcs.unittype.FlyingType attribute), 59  
height() (dcs.mapping.Rectangle method), 54  
Helicopter (class in dcs.unit), 56  
helicopter (dcs.unittype.FlyingType attribute), 59  
helicopter() (dcs.mission.Mission method), 11  
helicopter\_group() (dcs.mission.Mission method), 12  
helicopter\_num (dcs.unittype.ShipType attribute), 59  
HelicopterGroup (class in dcs.unitgroup), 58  
helicopters (dcs.country.Country attribute), 44  
Hold (class in dcs.task), 25

## I

Id (dcs.task.Aerobatics attribute), 25  
id (dcs.task.AFAC attribute), 29  
id (dcs.task.AntishipStrike attribute), 29  
Id (dcs.task.AntishipStrikeTaskAction attribute), 23  
Id (dcs.task.AttackGroup attribute), 23  
Id (dcs.task.AttackMapObject attribute), 23  
Id (dcs.task.AttackUnit attribute), 23  
id (dcs.task.AWACS attribute), 29  
Id (dcs.task.AWACSTaskAction attribute), 25  
Id (dcs.task.Bombing attribute), 24  
Id (dcs.task.BombingRunway attribute), 24  
id (dcs.task.CAP attribute), 30  
Id (dcs.task.CAP.EnrouteTasks.EngageGroup attribute), 30  
Id (dcs.task.CAP.EnrouteTasks.EngageTargets attribute), 30  
Id (dcs.task.CAP.EnrouteTasks.EngageTargetsInZone attribute), 30  
Id (dcs.task.CAP.EnrouteTasks.EngageUnit attribute), 30  
Id (dcs.task.CAPTTaskAction attribute), 24  
Id (dcs.task.CargoTransportation attribute), 27  
id (dcs.task.CAS attribute), 29  
Id (dcs.task.CAS.EnrouteTasks.EngageGroup attribute), 29  
Id (dcs.task.CAS.EnrouteTasks.EngageTargetsInZone attribute), 29  
Id (dcs.task.CAS.EnrouteTasks.EngageUnit attribute), 29  
Id (dcs.task.CASTaskAction attribute), 24  
Id (dcs.task.ControlledTask attribute), 19  
Id (dcs.task.DisembarkFromTransport attribute), 27  
Id (dcs.task.Embarking attribute), 26  
Id (dcs.task.EmbarkToTransport attribute), 26  
Id (dcs.task.EngageGroup attribute), 25  
Id (dcs.task.EngageTargets attribute), 24  
Id (dcs.task.EngageTargetsInZone attribute), 25  
Id (dcs.task.EngageUnit attribute), 25  
id (dcs.task.Escort attribute), 30  
Id (dcs.task.EscortTaskAction attribute), 24  
Id (dcs.task.FAC attribute), 26  
Id (dcs.task.FACEngageGroup attribute), 26  
id (dcs.task.FighterSweep attribute), 30  
Id (dcs.task.FighterSweepTaskAction attribute), 24  
Id (dcs.task.FireAtPoint attribute), 25  
Id (dcs.task.Follow attribute), 25  
id (dcs.task.GroundAttack attribute), 30  
Id (dcs.task.Hold attribute), 25  
id (dcs.task.Intercept attribute), 31  
Id (dcs.task.Land attribute), 26  
Id (dcs.task.NoTask attribute), 23  
id (dcs.task.Nothing attribute), 28  
Id (dcs.task.Option attribute), 32  
Id (dcs.task.OrbitAction attribute), 25  
id (dcs.task.PinpointStrike attribute), 31  
id (dcs.task.Reconnaissance attribute), 31  
id (dcs.task.Refueling attribute), 31  
Id (dcs.task.RefuelingTaskAction attribute), 25  
id (dcs.task.RunwayAttack attribute), 31  
id (dcs.task.SEAD attribute), 31  
Id (dcs.task.SEADTaskAction attribute), 24  
Id (dcs.task.Tanker attribute), 25  
id (dcs.task.Targets.All attribute), 20  
id (dcs.task.Targets.All.Air attribute), 20  
id (dcs.task.Targets.All.Air.Helicopters attribute), 21  
id (dcs.task.Targets.All.Air.Planes attribute), 20  
id (dcs.task.Targets.All.Air.Planes.Bombers attribute), 21  
id (dcs.task.Targets.All.Air.Planes.Fighters attribute), 21  
id (dcs.task.Targets.All.GroundUnits attribute), 21  
id (dcs.task.Targets.All.GroundUnits.AirDefence attribute), 21  
id (dcs.task.Targets.All.GroundUnits.AirDefence.AAA attribute), 22  
id (dcs.task.Targets.All.GroundUnits.AirDefence.AAA.SAMRelated attribute), 22  
id (dcs.task.Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.LRSA attribute), 22  
id (dcs.task.Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.MRSA attribute), 22  
id (dcs.task.Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.SRSA attribute), 22  
id (dcs.task.Targets.All.GroundUnits.Fortifications attribute), 21  
id (dcs.task.Targets.All.GroundUnits.GroundVehicles attribute), 21  
id (dcs.task.Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles attribute), 21  
id (dcs.task.Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.AP attribute), 21



- id (dcs.task.Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.IFV attribute), 21
  - id (dcs.task.Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.Tanks attribute), 21
  - id (dcs.task.Targets.All.GroundUnits.GroundVehicles.Artillery attribute), 21
  - id (dcs.task.Targets.All.GroundUnits.GroundVehicles.UnarmoredVehicles attribute), 21
  - id (dcs.task.Targets.All.GroundUnits.Infantry attribute), 21
  - id (dcs.task.Targets.All.Naval attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips.HeavyArmoredShips attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips.HeavyArmoredShips.AircraftCarriers attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips.HeavyArmoredShips.Corvettes attribute), 23
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips.HeavyArmoredShips.Cruisers attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips.HeavyArmoredShips.Destroyers attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips.HeavyArmoredShips.Frigates attribute), 22
  - id (dcs.task.Targets.All.Naval.Ships.ArmedShips.LightArmoredShips attribute), 23
  - id (dcs.task.Targets.All.Naval.Ships.UnarmedShips attribute), 23
  - id (dcs.task.Targets.All.Naval.Submarines attribute), 23
  - id (dcs.task.TargetType attribute), 20
  - id (dcs.task.Transport attribute), 31
  - Id (dcs.task.WrappedAction attribute), 27
  - id (dcs.terrain.caucasus.Anapa attribute), 35
  - id (dcs.terrain.caucasus.Batumi attribute), 38
  - id (dcs.terrain.caucasus.Beslan attribute), 40
  - id (dcs.terrain.caucasus.Gelendzhik attribute), 37
  - id (dcs.terrain.caucasus.Gudauta attribute), 37
  - id (dcs.terrain.caucasus.Kobuleti attribute), 38
  - id (dcs.terrain.caucasus.KrasnodarCenter attribute), 36
  - id (dcs.terrain.caucasus.KrasnodarPashkovsky attribute), 37
  - id (dcs.terrain.caucasus.Krymsk attribute), 36
  - id (dcs.terrain.caucasus.Kutaisi attribute), 38
  - id (dcs.terrain.caucasus.Lochini attribute), 39
  - id (dcs.terrain.caucasus.Maykop attribute), 36
  - id (dcs.terrain.caucasus.Mineralnye attribute), 39
  - id (dcs.terrain.caucasus.Mozdok attribute), 39
  - id (dcs.terrain.caucasus.Nalchik attribute), 39
  - id (dcs.terrain.caucasus.Novorossiysk attribute), 36
  - id (dcs.terrain.caucasus.Senaki attribute), 38
  - id (dcs.terrain.caucasus.Sochi attribute), 37
  - id (dcs.terrain.caucasus.Soganlug attribute), 40
  - id (dcs.terrain.caucasus.Vaziani attribute), 40
  - id (dcs.terrain.caucasus.Vladikavkaz attribute), 41
  - id (dcs.terrain.nevada.Groom attribute), 42
  - id (dcs.terrain.nevada.McCarran attribute), 42
  - id (dcs.terrain.nevada.Nellis attribute), 42
  - id (dcs.terrain.terrain.Airport attribute), 43
  - id (dcs.unittype.UnitType attribute), 59
  - in\_triangle() (dcs.mapping.Polygon static method), 54
  - IndicationTextEqual (class in dcs.goals), 48
  - Intercept (class in dcs.task), 30
  - is\_blue() (dcs.mission.Mission method), 17
  - is\_blue() (dcs.terrain.terrain.Airport method), 43
  - is\_clockwise() (dcs.mapping.Polygon method), 54
  - is\_index() (dcs.mapping.Polygon static method), 54
  - is\_red() (dcs.mission.Mission method), 17
  - is\_red() (dcs.terrain.terrain.Airport method), 43
  - Key (dcs.task.ActivateBeaconCommand attribute), 27
  - Key (dcs.task.AntishipStrikeTaskAction attribute), 23
  - Key (dcs.task.CAPTTaskAction attribute), 24
  - Key (dcs.task.CASTTaskAction attribute), 24
  - Key (dcs.task.EPLRS attribute), 27
  - Key (dcs.task.FighterSweepTaskAction attribute), 24
  - Key (dcs.task.OptDisperseUnderFire attribute), 33
  - Key (dcs.task.Option attribute), 32
  - Key (dcs.task.OptReactOnThreat attribute), 32
  - Key (dcs.task.OptROE attribute), 32
  - Key (dcs.task.RunScript attribute), 27
  - Key (dcs.task.RunScriptFile attribute), 27
  - Key (dcs.task.SEADTaskAction attribute), 24
  - Key (dcs.task.SetFrequencyCommand attribute), 28
  - Key (dcs.task.SetImmortalCommand attribute), 28
  - Key (dcs.task.SetInvisibleCommand attribute), 28
  - Key (dcs.task.StopTransmission attribute), 28
  - Key (dcs.task.SwitchWaypoint attribute), 28
  - Key (dcs.task.TransmitMessage attribute), 28
  - Kobuleti (class in dcs.terrain.caucasus), 38
  - kobuleti() (dcs.terrain.caucasus.Caucasus method), 41
  - krasnodar\_center() (dcs.terrain.caucasus.Caucasus method), 41
  - krasnodar\_pashkovsky() (dcs.terrain.caucasus.Caucasus method), 41
  - KrasnodarCenter (class in dcs.terrain.caucasus), 36
  - KrasnodarPashkovsky (class in dcs.terrain.caucasus), 37
  - Krymsk (class in dcs.terrain.caucasus), 36
  - krymsk() (dcs.terrain.caucasus.Caucasus method), 41
  - Kutaisi (class in dcs.terrain.caucasus), 38
  - kutaisi() (dcs.terrain.caucasus.Caucasus method), 41
- ## L
- Land (class in dcs.task), 26
  - land\_at() (dcs.unitgroup.FlyingGroup method), 58

languages() (dcs.translation.Translation method), 56  
large\_parking\_slot (dcs.unittypes.FlyingType attribute), 59  
length (dcs.unittypes.FlyingType attribute), 59  
load\_dict() (dcs.terrain.terrain.Warehouses method), 44  
load\_file() (dcs.mission.Mission method), 8  
load\_from\_dict() (dcs.forcedoptions.ForcedOptions method), 45  
load\_from\_dict() (dcs.goals.Goal method), 52  
load\_from\_dict() (dcs.goals.Goals method), 52  
load\_from\_dict() (dcs.groundcontrol.GroundControl method), 53  
load\_from\_dict() (dcs.mission.MapResource method), 17  
load\_from\_dict() (dcs.mission.Options method), 18  
load\_from\_dict() (dcs.point.MovingPoint method), 55  
load\_from\_dict() (dcs.point.PointProperties method), 55  
load\_from\_dict() (dcs.point.StaticPoint method), 54  
load\_from\_dict() (dcs.terrain.terrain.Airport method), 43  
load\_from\_dict() (dcs.terrain.terrain.MapView method), 43  
load\_from\_dict() (dcs.unit.FlyingUnit method), 56  
load\_from\_dict() (dcs.unit.Helicopter method), 56  
load\_from\_dict() (dcs.unit.Ship method), 56  
load\_from\_dict() (dcs.unit.Static method), 57  
load\_from\_dict() (dcs.unit.Unit method), 56  
load\_from\_dict() (dcs.unit.Vehicle method), 56  
load\_from\_dict() (dcs.unitgroup.FlyingGroup method), 58  
load\_from\_dict() (dcs.unitgroup.Group method), 57  
load\_from\_dict() (dcs.unitgroup.MovingGroup method), 57  
load\_from\_dict() (dcs.unitgroup.StaticGroup method), 58  
load\_from\_dict() (dcs.unitgroup.VehicleGroup method), 57  
load\_from\_dict() (dcs.weather.Weather method), 60  
load\_loadout() (dcs.unit.FlyingUnit method), 56  
load\_loadout() (dcs.unitgroup.FlyingGroup method), 58  
load\_loadout\_from\_file() (dcs.unitgroup.FlyingGroup method), 58  
load\_payloads() (dcs.unittypes.FlyingType class method), 60  
load\_pylon() (dcs.unit.FlyingUnit method), 56  
load\_pylon() (dcs.unitgroup.FlyingGroup method), 58  
load\_task\_default\_loadout() (dcs.unitgroup.FlyingGroup method), 58  
loadout() (dcs.unittypes.FlyingType class method), 60  
loadout\_by\_name() (dcs.unittypes.FlyingType class method), 60  
loads() (in module dcs.lua.parse), 35  
Lochini (class in dcs.terrain.caucasus), 39  
lochini() (dcs.terrain.caucasus.Caucasus method), 41

## M

MainTask (class in dcs.task), 28

map (dcs.task.MainTask attribute), 28  
map\_view\_default (dcs.terrain.caucasus.Caucasus attribute), 40  
map\_view\_default (dcs.terrain.nevada.Nevada attribute), 42  
map\_view\_default (dcs.terrain.terrain.Terrain attribute), 43  
MapResource (class in dcs.mission), 17  
MapView (class in dcs.terrain.terrain), 43  
mass (dcs.unittypes.StaticType attribute), 59  
max\_speed (dcs.unittypes.FlyingType attribute), 59  
Maykop (class in dcs.terrain.caucasus), 36  
maykop() (dcs.terrain.caucasus.Caucasus method), 41  
McCarran (class in dcs.terrain.nevada), 42  
mccarran() (dcs.terrain.nevada.Nevada method), 42  
Mineralnye (class in dcs.terrain.caucasus), 39  
mineralnye() (dcs.terrain.caucasus.Caucasus method), 41  
Mission (class in dcs.mission), 7  
MissionScoreHigher (class in dcs.goals), 48  
MissionScoreLower (class in dcs.goals), 48  
Modulation (class in dcs.task), 19  
MovingGroup (class in dcs.unitgroup), 57  
MovingPoint (class in dcs.point), 55  
Mozdok (class in dcs.terrain.caucasus), 39  
mozdok() (dcs.terrain.caucasus.Caucasus method), 41

## N

Nalchik (class in dcs.terrain.caucasus), 39  
nalchik() (dcs.terrain.caucasus.Caucasus method), 41  
name (dcs.task.AFAC attribute), 29  
name (dcs.task.AntishipStrike attribute), 29  
name (dcs.task.AWACS attribute), 29  
name (dcs.task.CAP attribute), 30  
name (dcs.task.CAS attribute), 29  
name (dcs.task.Escort attribute), 30  
name (dcs.task.FighterSweep attribute), 30  
name (dcs.task.GroundAttack attribute), 30  
name (dcs.task.Intercept attribute), 31  
name (dcs.task.MainTask attribute), 28  
name (dcs.task.Nothing attribute), 28  
name (dcs.task.PinpointStrike attribute), 31  
name (dcs.task.Reconnaissance attribute), 31  
name (dcs.task.Refueling attribute), 31  
name (dcs.task.RunwayAttack attribute), 31  
name (dcs.task.SEAD attribute), 31  
name (dcs.task.Transport attribute), 31  
name (dcs.terrain.caucasus.Anapa attribute), 35  
name (dcs.terrain.caucasus.Batumi attribute), 38  
name (dcs.terrain.caucasus.Beslan attribute), 40  
name (dcs.terrain.caucasus.Gelendzhik attribute), 37  
name (dcs.terrain.caucasus.Gudauta attribute), 38  
name (dcs.terrain.caucasus.Kobuleti attribute), 38  
name (dcs.terrain.caucasus.KrasnodarCenter attribute), 36

name (dcs.terrain.caucasus.KrasnodarPashkovsky attribute), 37

name (dcs.terrain.caucasus.Krymsk attribute), 36

name (dcs.terrain.caucasus.Kutaisi attribute), 38

name (dcs.terrain.caucasus.Lochini attribute), 39

name (dcs.terrain.caucasus.Maykop attribute), 36

name (dcs.terrain.caucasus.Mineralnye attribute), 39

name (dcs.terrain.caucasus.Mozdok attribute), 39

name (dcs.terrain.caucasus.Nalchik attribute), 39

name (dcs.terrain.caucasus.Novorossiysk attribute), 36

name (dcs.terrain.caucasus.Senaki attribute), 38

name (dcs.terrain.caucasus.Sochi attribute), 37

name (dcs.terrain.caucasus.Soganlug attribute), 40

name (dcs.terrain.caucasus.Sukhumi attribute), 37

name (dcs.terrain.caucasus.Vaziani attribute), 40

name (dcs.terrain.nevada.Creech attribute), 41

name (dcs.terrain.nevada.Groom attribute), 42

name (dcs.terrain.nevada.McCarran attribute), 42

name (dcs.terrain.nevada.Nellis attribute), 42

name (dcs.terrain.terrain.Airport attribute), 43

name (dcs.unittypes.UnitType attribute), 59

Nellis (class in dcs.terrain.nevada), 42

nellis() (dcs.terrain.nevada.Nevada method), 43

Nevada (class in dcs.terrain.nevada), 42

next\_callsign\_category() (dcs.country.Country method), 44

next\_callsign\_id() (dcs.country.Country method), 44

next\_dict\_id() (dcs.mission.Mission method), 9

next\_eplrs() (dcs.mission.Mission method), 9

next\_group\_id() (dcs.mission.Mission method), 8

next\_unit\_id() (dcs.mission.Mission method), 9

NoReaction (dcs.task.OptReactOnThreat.Values attribute), 32

NoTask (class in dcs.task), 23

Nothing (class in dcs.task), 28

Novorossiysk (class in dcs.terrain.caucasus), 36

novorossiysk() (dcs.terrain.caucasus.Caucasus method), 41

## O

OpenFire (dcs.task.OptROE.Values attribute), 32

OpenFireWeaponFree (dcs.task.OptROE.Values attribute), 32

OptDisperseUnderFire (class in dcs.task), 33

Option (class in dcs.task), 32

Options (class in dcs.mission), 18

OptReactOnThreat (class in dcs.task), 32

OptReactOnThreat.Values (class in dcs.task), 32

OptROE (class in dcs.task), 32

OptROE.Values (class in dcs.task), 32

Or (class in dcs.goals), 48

OrbitAction (class in dcs.task), 25

OrbitAction.OrbitPattern (class in dcs.task), 25

outbound\_rectangle() (dcs.mapping.Polygon method), 54

## P

panel\_radio (dcs.unittypes.FlyingType attribute), 60

parking (dcs.unittypes.ShipType attribute), 59

ParkingSlot (class in dcs.terrain.terrain), 43

PartOfCoalitionInZone (class in dcs.goals), 48

PartOfCoalitionOutsideZone (class in dcs.goals), 49

PartOfGroupInZone (class in dcs.goals), 49

PartOfGroupOutsideZone (class in dcs.goals), 49

PassiveDefense (dcs.task.OptReactOnThreat.Values attribute), 32

patriot\_site() (dcs.templates.VehicleTemplate.USA static method), 55

patrol\_flight() (dcs.mission.Mission method), 16

payload\_dirs (dcs.unittypes.FlyingType attribute), 60

payloads (dcs.unittypes.FlyingType attribute), 60

perform\_task (dcs.task.AntishipStrike attribute), 29

perform\_task (dcs.task.AWACS attribute), 29

perform\_task (dcs.task.CAP attribute), 30

perform\_task (dcs.task.CAS attribute), 29

perform\_task (dcs.task.Escort attribute), 30

perform\_task (dcs.task.FighterSweep attribute), 30

perform\_task (dcs.task.MainTask attribute), 28

perform\_task (dcs.task.Reconnaissance attribute), 31

perform\_task (dcs.task.Refueling attribute), 31

perform\_task (dcs.task.RunwayAttack attribute), 31

perform\_task (dcs.task.SEAD attribute), 31

perform\_task (dcs.task.Transport attribute), 32

PinpointStrike (class in dcs.task), 31

Plane (class in dcs.unit), 56

plane() (dcs.mission.Mission method), 11

plane\_group() (dcs.mission.Mission method), 11

plane\_num (dcs.unittypes.ShipType attribute), 59

PlaneGroup (class in dcs.unitgroup), 58

planes (dcs.country.Country attribute), 44

PlayerScoreLess (class in dcs.goals), 49

PlayerScoreMore (class in dcs.goals), 49

Point (class in dcs.mapping), 53

point\_from\_heading() (dcs.mapping.Point method), 53

point\_from\_heading() (in module dcs.mapping), 53

point\_in\_poly() (dcs.mapping.Polygon method), 54

point\_in\_rect() (dcs.mapping.Rectangle method), 54

PointAction (class in dcs.point), 54

PointProperties (class in dcs.point), 55

Polygon (class in dcs.mapping), 54

position (dcs.terrain.caucasus.Anapa attribute), 35

position (dcs.terrain.caucasus.Batumi attribute), 38

position (dcs.terrain.caucasus.Beslan attribute), 40

position (dcs.terrain.caucasus.Gelendzhik attribute), 37

position (dcs.terrain.caucasus.Gudauta attribute), 38

position (dcs.terrain.caucasus.Kobuleti attribute), 38

position (dcs.terrain.caucasus.KrasnodarCenter attribute), 36

position (dcs.terrain.caucasus.KrasnodarPashkovsky attribute), 37

position (dcs.terrain.caucasus.Krymsk attribute), 36  
position (dcs.terrain.caucasus.Kutaisi attribute), 39  
position (dcs.terrain.caucasus.Lochini attribute), 39  
position (dcs.terrain.caucasus.Maykop attribute), 36  
position (dcs.terrain.caucasus.Mineralnye attribute), 39  
position (dcs.terrain.caucasus.Mozdok attribute), 39  
position (dcs.terrain.caucasus.Nalchik attribute), 39  
position (dcs.terrain.caucasus.Novorossiysk attribute), 36  
position (dcs.terrain.caucasus.Senaki attribute), 38  
position (dcs.terrain.caucasus.Sochi attribute), 37  
position (dcs.terrain.caucasus.Soganlug attribute), 40  
position (dcs.terrain.caucasus.Sukhumi attribute), 37  
position (dcs.terrain.caucasus.Vaziani attribute), 40  
position (dcs.terrain.nevada.Creech attribute), 41  
position (dcs.terrain.nevada.Groom attribute), 42  
position (dcs.terrain.nevada.McCarran attribute), 42  
position (dcs.terrain.nevada.Nellis attribute), 42  
position (dcs.terrain.terrain.Airport attribute), 43  
position (dcs.unitgroup.Group attribute), 57  
Predicate (class in dcs.goals), 49  
predicate (dcs.goals.AllOfCoalitionInZone attribute), 45  
predicate (dcs.goals.AllOfCoalitionOutsideZone attribute), 45  
predicate (dcs.goals.AllOfGroupInZone attribute), 45  
predicate (dcs.goals.AllOfGroupOutsideZone attribute), 45  
predicate (dcs.goals.ArgumentInRange attribute), 45  
predicate (dcs.goals.BombInZone attribute), 46  
predicate (dcs.goals.CargoUnhookedInZone attribute), 46  
predicate (dcs.goals.CoalitionHasAirdrome attribute), 46  
predicate (dcs.goals.CoalitionHasHelipad attribute), 46  
predicate (dcs.goals.CockpitHighlightVisible attribute), 46  
predicate (dcs.goals.CockpitParamEqual attribute), 46  
predicate (dcs.goals.CockpitParamEqualToAnother attribute), 46  
predicate (dcs.goals.CockpitParamInRange attribute), 46  
predicate (dcs.goals.FlagEquals attribute), 47  
predicate (dcs.goals.FlagEqualsFlag attribute), 47  
predicate (dcs.goals.FlagIsFalse attribute), 47  
predicate (dcs.goals.FlagIsLess attribute), 47  
predicate (dcs.goals.FlagIsLessThanFlag attribute), 47  
predicate (dcs.goals.FlagIsMore attribute), 47  
predicate (dcs.goals.FlagIsTrue attribute), 47  
predicate (dcs.goals.GroupAlive attribute), 48  
predicate (dcs.goals.GroupDead attribute), 48  
predicate (dcs.goals.GroupLifeLess attribute), 48  
predicate (dcs.goals.IndicationTextEqual attribute), 48  
predicate (dcs.goals.MissionScoreHigher attribute), 48  
predicate (dcs.goals.MissionScoreLower attribute), 48  
predicate (dcs.goals.Or attribute), 48  
predicate (dcs.goals.PartOfCoalitionInZone attribute), 48  
predicate (dcs.goals.PartOfCoalitionOutsideZone attribute), 49

predicate (dcs.goals.PartOfGroupInZone attribute), 49  
predicate (dcs.goals.PartOfGroupOutsideZone attribute), 49  
predicate (dcs.goals.PlayerScoreLess attribute), 49  
predicate (dcs.goals.PlayerScoreMore attribute), 49  
predicate (dcs.goals.Predicate attribute), 49  
predicate (dcs.goals.Random attribute), 49  
predicate (dcs.goals.SignalFlareInZone attribute), 49  
predicate (dcs.goals.TimeAfter attribute), 50  
predicate (dcs.goals.TimeBefore attribute), 50  
predicate (dcs.goals.TimeSinceFlag attribute), 50  
predicate (dcs.goals.UnitAlive attribute), 50  
predicate (dcs.goals.UnitAltitudeHigher attribute), 50  
predicate (dcs.goals.UnitAltitudeLower attribute), 50  
predicate (dcs.goals.UnitBankWithin attribute), 50  
predicate (dcs.goals.UnitDamaged attribute), 50  
predicate (dcs.goals.UnitDead attribute), 51  
predicate (dcs.goals.UnitHeadingWithin attribute), 51  
predicate (dcs.goals.UnitInMovingZone attribute), 51  
predicate (dcs.goals.UnitInZone attribute), 51  
predicate (dcs.goals.UnitLifeLess attribute), 51  
predicate (dcs.goals.UnitOutsideMovingZone attribute), 51  
predicate (dcs.goals.UnitOutsideZone attribute), 51  
predicate (dcs.goals.UnitPitchWithin attribute), 52  
predicate (dcs.goals.UnitSpeedHigher attribute), 52  
predicate (dcs.goals.UnitSpeedLower attribute), 52  
predicate (dcs.goals.UnitVerticalSpeedWithin attribute), 52  
print\_stats() (dcs.mission.Mission method), 17  
pylons (dcs.unittypes.FlyingType attribute), 60

## R

radio\_frequency (dcs.unittypes.FlyingType attribute), 60  
Random (class in dcs.goals), 49  
random\_distant\_points() (dcs.mapping.Rectangle method), 54  
random\_point() (dcs.mapping.Polygon method), 54  
random\_point() (dcs.mapping.Rectangle method), 54  
random\_point() (dcs.mapping.Triangle method), 54  
random\_unit\_zone() (dcs.terrain.terrain.Airport method), 43  
rate (dcs.unittypes.StaticType attribute), 59  
Reconnaissance (class in dcs.task), 31  
Rectangle (class in dcs.mapping), 54  
refuel\_flight() (dcs.mission.Mission method), 15  
Refueling (class in dcs.task), 31  
RefuelingTaskAction (class in dcs.task), 25  
reload() (dcs.mission.Mission method), 17  
reset\_loadout() (dcs.unit.FlyingUnit method), 56  
reset\_loadout() (dcs.unitgroup.FlyingGroup method), 58  
resize() (dcs.mapping.Rectangle method), 54  
ReturnFire (dcs.task.OptROE.Values attribute), 32  
RunScript (class in dcs.task), 27



RunScriptFile (class in dcs.task), 27  
 Runway (class in dcs.terrain.terrain), 43  
 Runway (dcs.mission.StartType attribute), 7  
 RunwayAttack (class in dcs.task), 31

## S

sa10\_site() (dcs.templates.VehicleTemplate.Russia static method), 55  
 save() (dcs.mission.Mission method), 17  
 Scale (class in dcs.point), 54  
 scan\_payload\_dir() (dcs.unittypes.FlyingType class method), 60  
 sea\_object (dcs.unittypes.StaticType attribute), 59  
 SEAD (class in dcs.task), 31  
 SEADTaskAction (class in dcs.task), 24  
 season\_from\_start\_time (dcs.mission.Mission attribute), 7  
 Senaki (class in dcs.terrain.caucasus), 38  
 senaki() (dcs.terrain.caucasus.Caucasus method), 41  
 set() (dcs.translation.String method), 55  
 set\_blue() (dcs.terrain.terrain.Airport method), 43  
 set\_client() (dcs.unit.FlyingUnit method), 56  
 set\_client() (dcs.unitgroup.FlyingGroup method), 58  
 set\_coalition() (dcs.terrain.terrain.Airport method), 43  
 set\_default\_preset\_channel() (dcs.unit.FlyingUnit method), 56  
 set\_description\_bluetask\_text() (dcs.mission.Mission method), 8  
 set\_description\_redtask\_text() (dcs.mission.Mission method), 8  
 set\_description\_text() (dcs.mission.Mission method), 8  
 set\_frequency() (dcs.unitgroup.FlyingGroup method), 58  
 set\_neutral() (dcs.terrain.terrain.Airport method), 43  
 set\_parking() (dcs.unit.FlyingUnit method), 56  
 set\_player() (dcs.unit.FlyingUnit method), 56  
 set\_radio\_preset() (dcs.unit.FlyingUnit method), 56  
 set\_red() (dcs.terrain.terrain.Airport method), 43  
 set\_season\_from\_datetime() (dcs.weather.Weather method), 60  
 set\_skill() (dcs.unitgroup.Group method), 57  
 set\_sortie\_text() (dcs.mission.Mission method), 8  
 set\_string() (dcs.translation.Translation method), 55  
 SetFrequencyCommand (class in dcs.task), 28  
 SetImmortalCommand (class in dcs.task), 28  
 SetInvisibleCommand (class in dcs.task), 28  
 shape\_name (dcs.unittypes.StaticType attribute), 59  
 Ship (class in dcs.unit), 56  
 ship() (dcs.mission.Mission method), 10  
 ship\_group() (dcs.mission.Mission method), 10  
 ShipGroup (class in dcs.unitgroup), 58  
 ShipType (class in dcs.unittypes), 59  
 SignalFlareInZone (class in dcs.goals), 49  
 Skill (class in dcs.unit), 56  
 slot\_version (dcs.terrain.nevada.Creech attribute), 41  
 slot\_version (dcs.terrain.nevada.Groom attribute), 42  
 slot\_version (dcs.terrain.nevada.McCarran attribute), 42  
 slot\_version (dcs.terrain.nevada.Nellis attribute), 42  
 slot\_version (dcs.terrain.terrain.Airport attribute), 43  
 Sochi (class in dcs.terrain.caucasus), 37  
 sochi() (dcs.terrain.caucasus.Caucasus method), 41  
 Soganlug (class in dcs.terrain.caucasus), 40  
 soganlug() (dcs.terrain.caucasus.Caucasus method), 40  
 sortie\_text() (dcs.mission.Mission method), 8  
 Spring (dcs.weather.Weather.Season attribute), 60  
 start\_after\_time() (dcs.task.ControlledTask method), 19  
 start\_if\_lua\_predicate() (dcs.task.ControlledTask method), 20  
 start\_if\_user\_flag() (dcs.task.ControlledTask method), 19  
 start\_probability() (dcs.task.ControlledTask method), 20  
 starts\_from\_airport() (dcs.unitgroup.FlyingGroup method), 57  
 StartType (class in dcs.mission), 7  
 Static (class in dcs.unit), 57  
 static() (dcs.mission.Mission method), 9  
 static\_group() (dcs.mission.Mission method), 9  
 StaticGroup (class in dcs.unitgroup), 58  
 StaticPoint (class in dcs.point), 54  
 StaticType (class in dcs.unittypes), 59  
 stats() (dcs.mission.Mission method), 17  
 Steer (class in dcs.point), 55  
 stop\_after\_duration() (dcs.task.ControlledTask method), 20  
 stop\_after\_time() (dcs.task.ControlledTask method), 20  
 stop\_if\_lua\_predicate() (dcs.task.ControlledTask method), 20  
 stop\_if\_user\_flag() (dcs.task.ControlledTask method), 20  
 StopTransmission (class in dcs.task), 28  
 store() (dcs.mission.MapResource method), 18  
 store\_loadout() (dcs.unit.FlyingUnit method), 56  
 str() (dcs.translation.String method), 55  
 String (class in dcs.translation), 55  
 string() (dcs.mission.Mission method), 9  
 sub\_tasks (dcs.task.AFAC attribute), 29  
 sub\_tasks (dcs.task.AntishipStrike attribute), 29  
 sub\_tasks (dcs.task.AWACS attribute), 29  
 sub\_tasks (dcs.task.CAP attribute), 30  
 sub\_tasks (dcs.task.CAS attribute), 29  
 sub\_tasks (dcs.task.Escort attribute), 30  
 sub\_tasks (dcs.task.FighterSweep attribute), 30  
 sub\_tasks (dcs.task.GroundAttack attribute), 30  
 sub\_tasks (dcs.task.Intercept attribute), 31  
 sub\_tasks (dcs.task.MainTask attribute), 28  
 sub\_tasks (dcs.task.Nothing attribute), 29  
 sub\_tasks (dcs.task.PinpointStrike attribute), 31  
 sub\_tasks (dcs.task.Reconnaissance attribute), 31  
 sub\_tasks (dcs.task.Refueling attribute), 31  
 sub\_tasks (dcs.task.RunwayAttack attribute), 31  
 sub\_tasks (dcs.task.SEAD attribute), 31

sub\_tasks (dcs.task.Transport attribute), 31  
Sukhumi (class in dcs.terrain.caucasus), 37  
sukhumi() (dcs.terrain.caucasus.Caucasus method), 41  
Summer (dcs.weather.Weather.Season attribute), 60  
SwitchWaypoint (class in dcs.task), 28

## T

tacan (dcs.terrain.caucasus.Anapa attribute), 35  
tacan (dcs.terrain.caucasus.Batumi attribute), 38  
tacan (dcs.terrain.caucasus.Beslan attribute), 40  
tacan (dcs.terrain.caucasus.Gelendzhik attribute), 37  
tacan (dcs.terrain.caucasus.Gudauta attribute), 38  
tacan (dcs.terrain.caucasus.Kobuleti attribute), 38  
tacan (dcs.terrain.caucasus.KrasnodarCenter attribute), 36  
tacan (dcs.terrain.caucasus.KrasnodarPashkovsky attribute), 37  
tacan (dcs.terrain.caucasus.Krymsk attribute), 36  
tacan (dcs.terrain.caucasus.Kutaisi attribute), 39  
tacan (dcs.terrain.caucasus.Lochini attribute), 40  
tacan (dcs.terrain.caucasus.Maykop attribute), 36  
tacan (dcs.terrain.caucasus.Mineralnye attribute), 39  
tacan (dcs.terrain.caucasus.Mozdok attribute), 39  
tacan (dcs.terrain.caucasus.Nalchik attribute), 39  
tacan (dcs.terrain.caucasus.Novorossiysk attribute), 36  
tacan (dcs.terrain.caucasus.Senaki attribute), 38  
tacan (dcs.terrain.caucasus.Sochi attribute), 37  
tacan (dcs.terrain.caucasus.Soganlug attribute), 40  
tacan (dcs.terrain.caucasus.Sukhumi attribute), 37  
tacan (dcs.terrain.caucasus.Vaziani attribute), 40  
tacan (dcs.terrain.nevada.Creech attribute), 41  
tacan (dcs.terrain.nevada.Groom attribute), 42  
tacan (dcs.terrain.nevada.McCarran attribute), 42  
tacan (dcs.terrain.nevada.Nellis attribute), 42  
tacan (dcs.terrain.terrain.Airport attribute), 43  
tacan (dcs.unittypes.FlyingType attribute), 60  
Tanker (class in dcs.task), 25  
Targets (class in dcs.task), 20  
Targets.All (class in dcs.task), 20  
Targets.All.Air (class in dcs.task), 20  
Targets.All.Air.Helicopters (class in dcs.task), 21  
Targets.All.Air.Planes (class in dcs.task), 20  
Targets.All.Air.Planes.Bombers (class in dcs.task), 21  
Targets.All.Air.Planes.Fighters (class in dcs.task), 20  
Targets.All.GroundUnits (class in dcs.task), 21  
Targets.All.GroundUnits.AirDefence (class in dcs.task), 21  
Targets.All.GroundUnits.AirDefence.AAA (class in dcs.task), 22  
Targets.All.GroundUnits.AirDefence.AAA.SAMRelated (class in dcs.task), 22  
Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.LRSAM (class in dcs.task), 22  
Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.MRSAM (class in dcs.task), 22  
Targets.All.GroundUnits.AirDefence.AAA.SAMRelated.SRSAM (class in dcs.task), 22  
Targets.All.GroundUnits.Fortifications (class in dcs.task), 21  
Targets.All.GroundUnits.GroundVehicles (class in dcs.task), 21  
Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles (class in dcs.task), 21  
Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.APC (class in dcs.task), 21  
Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.IFV (class in dcs.task), 21  
Targets.All.GroundUnits.GroundVehicles.ArmoredVehicles.Tanks (class in dcs.task), 21  
Targets.All.GroundUnits.GroundVehicles.Artilletry (class in dcs.task), 21  
Targets.All.GroundUnits.GroundVehicles.UnarmedVehicles (class in dcs.task), 21  
Targets.All.GroundUnits.Infantry (class in dcs.task), 21  
Targets.All.Naval (class in dcs.task), 22  
Targets.All.Naval.Ships (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.AircraftCarriers (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Corvettes (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Cruisers (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Destroyers (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips.HeavyArmedShips.Frigates (class in dcs.task), 22  
Targets.All.Naval.Ships.ArmedShips.LightArmedShips (class in dcs.task), 23  
Targets.All.Naval.Ships.UnarmedShips (class in dcs.task), 23  
Targets.All.Naval.Submarines (class in dcs.task), 23  
TargetType (class in dcs.task), 20  
Task (class in dcs.task), 19  
task\_default (dcs.unittypes.FlyingType attribute), 60  
tasks (dcs.unittypes.FlyingType attribute), 60  
Terrain (class in dcs.terrain.terrain), 43  
TimeAfter (class in dcs.goals), 50  
TimeBefore (class in dcs.goals), 50  
TimeSinceFlag (class in dcs.goals), 50  
Translation (class in dcs.translation), 55  
TransmitMessage (class in dcs.task), 27  
Transport (class in dcs.task), 31  
Triangle (class in dcs.mapping), 53

triangulate() (dcs.mapping.Polygon method), 54

## U

Unit (class in dcs.unit), 56

unit\_zones (dcs.terrain.caucasus.Anapa attribute), 35

unit\_zones (dcs.terrain.caucasus.Batumi attribute), 38

unit\_zones (dcs.terrain.caucasus.Beslan attribute), 40

unit\_zones (dcs.terrain.caucasus.Gelendzhik attribute), 37

unit\_zones (dcs.terrain.caucasus.Gudauta attribute), 38

unit\_zones (dcs.terrain.caucasus.Kobuleti attribute), 38

unit\_zones (dcs.terrain.caucasus.KrasnodarCenter attribute), 36

unit\_zones (dcs.terrain.caucasus.KrasnodarPashkovsky attribute), 37

unit\_zones (dcs.terrain.caucasus.Krymsk attribute), 36

unit\_zones (dcs.terrain.caucasus.Kutaisi attribute), 39

unit\_zones (dcs.terrain.caucasus.Lochini attribute), 40

unit\_zones (dcs.terrain.caucasus.Maykop attribute), 36

unit\_zones (dcs.terrain.caucasus.Mineralnye attribute), 39

unit\_zones (dcs.terrain.caucasus.Mozdok attribute), 39

unit\_zones (dcs.terrain.caucasus.Nalchik attribute), 39

unit\_zones (dcs.terrain.caucasus.Novorossiysk attribute), 36

unit\_zones (dcs.terrain.caucasus.Senaki attribute), 38

unit\_zones (dcs.terrain.caucasus.Sochi attribute), 37

unit\_zones (dcs.terrain.caucasus.Soganlug attribute), 40

unit\_zones (dcs.terrain.caucasus.Sukhumi attribute), 37

unit\_zones (dcs.terrain.caucasus.Vaziani attribute), 40

unit\_zones (dcs.terrain.nevada.Creech attribute), 41

unit\_zones (dcs.terrain.nevada.Groom attribute), 42

unit\_zones (dcs.terrain.nevada.McCarran attribute), 42

unit\_zones (dcs.terrain.nevada.Nellis attribute), 42

unit\_zones (dcs.terrain.terrain.Airport attribute), 43

UnitAlive (class in dcs.goals), 50

UnitAltitudeHigher (class in dcs.goals), 50

UnitAltitudeLower (class in dcs.goals), 50

UnitBankWithin (class in dcs.goals), 50

UnitDamaged (class in dcs.goals), 50

UnitDead (class in dcs.goals), 51

UnitHeadingWithin (class in dcs.goals), 51

UnitInMovingZone (class in dcs.goals), 51

UnitInZone (class in dcs.goals), 51

UnitLifeLess (class in dcs.goals), 51

UnitOutsideMovingZone (class in dcs.goals), 51

UnitOutsideZone (class in dcs.goals), 51

UnitPitchWithin (class in dcs.goals), 51

UnitSpeedHigher (class in dcs.goals), 52

UnitSpeedLower (class in dcs.goals), 52

UnitType (class in dcs.unittypes), 59

UnitVerticalSpeedWithin (class in dcs.goals), 52

## V

Vaziani (class in dcs.terrain.caucasus), 40

vaziani() (dcs.terrain.caucasus.Caucasus method), 41

Vehicle (class in dcs.unit), 56

vehicle() (dcs.mission.Mission method), 10

vehicle\_group() (dcs.mission.Mission method), 10

vehicle\_group\_platoon() (dcs.mission.Mission method), 10

VehicleGroup (class in dcs.unitgroup), 57

VehicleTemplate (class in dcs.templates), 55

VehicleTemplate.Russia (class in dcs.templates), 55

VehicleTemplate.USA (class in dcs.templates), 55

VehicleType (class in dcs.unittypes), 59

VNav (class in dcs.point), 54

## W

Warehouses (class in dcs.terrain.terrain), 44

Warm (dcs.mission.StartType attribute), 7

WeaponFree (dcs.task.OptROE.Values attribute), 32

WeaponHold (dcs.task.OptROE.Values attribute), 32

WeaponType (class in dcs.task), 20

Weather (class in dcs.weather), 60

Weather.BarcSystem (class in dcs.weather), 60

Weather.Season (class in dcs.weather), 60

width (dcs.unittypes.FlyingType attribute), 59

width() (dcs.mapping.Rectangle method), 54

Wind (class in dcs.weather), 60

Winter (dcs.weather.Weather.Season attribute), 60

WrappedAction (class in dcs.task), 27

## X

x (dcs.unitgroup.Group attribute), 57

## Y

y (dcs.unitgroup.Group attribute), 57