Grand Challenge

**SoSID Toolkit Manual**

Author(s): Nikolaos Kalliatakis, Hussain Nabih Naeem, Nazlican Cigal

Grant Agreement number: 101097120

Project acronym: COLOSSUS

Project title: Collaborative System of Systems Exploration of Aviation Products, Services and Business Models

Start date of the project: 01/02/2023

Duration: 36 months

Project coordinator name & organisation:

Prajwal Shiva Prakasha

DLR - German Aerospace Center Institute of System Architectures in Aeronautics | Aviation System Design and Assessment

Tel: +49 40 2489641-322

E-mail: [prajwal.prakasha@dlr.de](mailto:prajwal.prakasha@dlr.de)

Project website address: https://colossus-sos-project.eu/

TABLE OF CONTENTS

1. Terms of Use 3

2. Wildfire Simulation 5

2.1 Basics of the Simulation 5

2.2 Running the Simulation 5

2.3 Wildfire Specific Parameters 6

2.4 Terrain Specific Parameters 9

2.5 Atmosphere Specific Parameters 10

2.6 Aircraft Agent Specific Parameters 11

2.6.1 Aircraft propulsion input parameters 12

2.6.2 Aircraft profile input parameters 16

2.7 Suppression Tactic Parameters 18

2.7.1 SuppressionTacticInput Options 18

2.7.2 *change\_condition* Options 19

List of figures and tables

[Table 1. Wildfire Parameter Inputs 6](#_Toc190685925)

[Table 2. Terrain Parameter Inputs 9](#_Toc190685926)

[Table 3. Atmosphere Parameter Inputs 10](#_Toc190685927)

[Table 4 Wildfire Aircraft Agent Parameter Inputs 11](#_Toc190685928)

[Table 5 Aircraft Propulsion Input Parameters 12](#_Toc190685929)

[Table 6 Aircraft Mission Profile Input Parameters 16](#_Toc190685930)

[Table 7 Suppression Tactic Parameters 18](#_Toc190685931)

# Terms of Use

Terms of Use for SoSID Toolkit (Beta Version)

1. Acceptance of Terms
   1. By accessing, downloading, installing, or using the SoSID Toolkit, you agree to be bound by these Terms of Use ("Terms"). If you do not agree to these Terms, please do not use SoSID Toolkit. These Terms apply to all users of the software, including those who are also contributors of content, information, and other materials or services on the Website.
2. Beta Version Disclaimer
   1. The SoSID Toolkit you are about to use is released in its beta version. It is still under development, and its features and functionalities are not fully complete. The purpose of this beta release is to obtain feedback from users like you about the software's performance and usability.
   2. As a beta version, please be advised that the results produced by the SoSID Toolkit are preliminary and may be subject to errors or inaccuracies. Users are cautioned to use the software for research purposes only and not for critical operations.
   3. Your use of this software indicates your understanding and acceptance of the risks associated with beta software.
3. Reporting of Bugs and Strange Behaviors
   1. Users are highly encouraged to report any bugs, errors, or strange behaviors encountered while using the SoSID Toolkit. Your feedback is invaluable in improving the software and resolving issues.
   2. Please submit reports via email to one of the associated emails with a detailed description of the issue, including steps to reproduce it, screenshots if applicable, and any other information that might be helpful in diagnosing the problem.
4. Publications and Dissemination
   1. The use of the SoSID Toolkit for research that results in publications, presentations, or any form of dissemination is subject to the following conditions:
      1. All such activities must be discussed with and approved by the developers of the SoSID Toolkit prior to public disclosure. This allows us to ensure the accuracy of the information being disseminated and to properly acknowledge the contributions of the software to your research.
      2. Requests for approval should be sent to [prajwal.prakasha@dlr.de](mailto:prajwal.prakasha@dlr.de), including details of the publication or dissemination activity.
   2. The purpose of this condition is to foster collaborative research and to ensure that the use of the SoSID Toolkit in public domains accurately reflects its capabilities and limitations.
5. General Conditions
   1. The SoSID Toolkit is provided "as is", and the developers make no warranties regarding its performance, reliability, or suitability for any particular purpose. By using this software, you agree to do so at your own risk.
   2. The developers reserve the right to modify these Terms of Use at any time without notice. Your continued use of the SoSID Toolkit after any such changes constitutes your acceptance of the new Terms.
   3. These Terms constitute the entire agreement between you and the developers of the SoSID Toolkit and govern your use of the software, superseding any prior agreements between you and the developers.

Contact Information

For any questions, concerns, or suggestions regarding these Terms of Use, please contact one of the associated developers:

Hussain Nabih Naeem – [Nabih.naeem@dlr.de](mailto:Nabih.naeem@dlr.de)

Nazlican Cigal – [Nazlican.Cigal@dlr.de](mailto:Nazlican.Cigal@dlr.de)

Nikolaos Kalliatakis – [Nikolaos.kalliatakis@dlr.de](mailto:Nikolaos.kalliatakis@dlr.de)

# Wildfire Simulation

## Basics of the Simulation

The wildfire simulation utilizes two main models: a fire model based on cellular automata and an agent-based model which represents the system of systems (SoS) of the wildfire fighting forces. At each time step in the simulation, the fire model computes its spread across a terrain and then aircraft agents (and soon ground agents) will determine how to approach and suppress the fire progression based on an operational doctrine. There are many variables that affect how this procedure is conducted and its success, from simulation parameters, to operational parameters and agent-based designs. These are all tune-able based on an input file.

At the current state, certain parameters such as the terrain and mission start time are fixed and unchangeable. This is to ensure that the scenario descriptions outlined in the project scenario are maintained to a consistent degree. The terrain features, such as the vegetation, residential areas, water areas and elevation data are all fixed to the associated map region. For the Grand Challenge, nearly all inputs in the scenarios are fixed as teams are encouraged to design their own aircraft and define new `agents` in the input file. In addition to this, part of the challenge is the exploration of SoS. As such, it is expected that even though the scenarios are pre-defined, changes to operational considerations are acceptable (so long as adequate reasoning and argumentation is provided). It is also expected that new inputs may be defined in the scenario input files, based on any developments made by the teams.

Control and manipulation of the wildfire simulation inputs is done based on an input file `*toolkit\_input.json`*. Within this file, there is a possibility to refer to other files, namely the agent definition files (to be used for defining the aircraft designs). **Unless specified, all inputs will be assumed as default values**.

## Running the Simulation

1. Upon copying and installing the repository, read and follow the instructions in the `*CONTRIBUTING.md`*
2. Navigate to the *examples/wildfire/data/scenarios/inputs* folder to view the different scenarios provided in the Grand Challenge.
3. To apply a certain scenario, navigate to *examples/wildfire/main.py* and change the input file path matching to the scenario you would like to run. If you would like to make changes to the input file **it is recommended to make a copy of the input file, apply any changes, and then place the copied file path into the *`Overwrites`*.** Doing so will ensure that the default values for the scenario are maintained and any additional changes are applied afterwards.
4. To change the agents/ aircraft, change the *`agents`* input in the overwrites/ copied input file matching the aircraft file path present in *examples/wildfire/data/aircraft/*.
5. Run *`main.py`* via the terminal or the coding software.
6. A GUI of the wildfire simulation should open. Run the simulation by clicking `start` in the bottom left corner, see Figure 1.

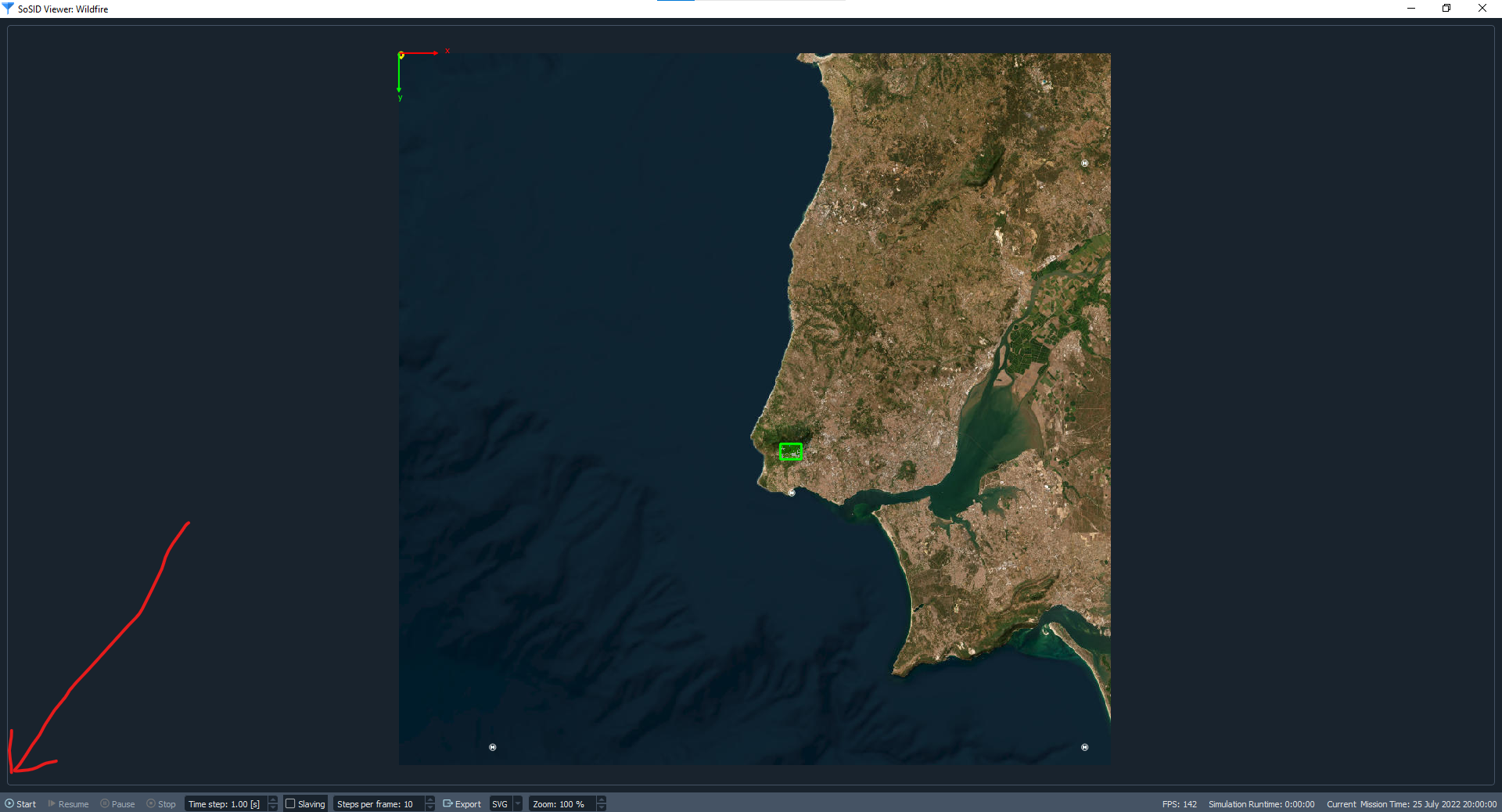


Figure 1 GUI of SoSID Toolkit

1. Once completed or stopped, close the GUI of the simulation. All simulation outputs will be written to in the output files present in *examples/wildfire/data/scenarios/outputs*

Below, the various inputs across the scenario input files and aircraft input files are described. Please refer to these tables to understand what the parameters mean.

Some parameters may be highlighted to indicate that they are under **no conditions modifiable** in the Grand Challenge.

The other parameters are modifiable given sufficient reasoning. For example, once could add airbases if their SoS design idea is based on having a DIY airbase which can be quickly set-up in a new location (potentially for smaller aircraft/ eVTOLS). Video guides may assist you as well in navigating how to change these parameters and the code base, and are present in the resources folder.

## Wildfire Specific Parameters

These are the bulk inputs of the wildfire simulation. All subsidiary inputs, such as terrain specifications, atmosphere specifications and agent specifications (for aircraft design) are referred to in this class.

Table 1. Wildfire Parameter Inputs

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Definition** | **Format [unit]** | **Default values**  **[comments]** |
| *time\_step* | The smallest time unit for each simulation iteration step for the agent model | float [s] | 1 |
| *enable\_adaptive\_time\_step* | Whether the wildfire model should calculate its own ideal timestep. Setting to *false* will use the *time\_step* for the fire model as well. | bool | true |
| *max\_runtime* | The maximum number of seconds from mission start to simulate | int [s] | 43200 |
| *output\_sampling\_time* | Frequency of output generation for results | int [s] | 300 |
| *map\_in\_map* | Whether the operational (larger) map should be utilized | bool | true |
| *export\_img* | Whether an image of the end state of the fire map should be outputted | bool | true |
| *import\_features\_osm* | Whether to use the OSM defined terrain features | bool | true |
| *ignition\_centers* | The location of ignition centers in GPS coordinates or grid indices. | List(dict{  “gps\_coords” : (lat,lon)})  [lat,lon] | [{“gps\_coords”:  [38.76, 9.44]}, {“gps\_coords” :  [38.6, 9.04]}] |
| *correction\_coefficient* | A factor applied in the wildfire model to better simulate fires | float | 0.4 |
| *deploy\_osm\_airport\_locations* | Whether OSM obtained airport data should be employed. | bool | true |
| *k\_nearest\_airports* | The number of OSM found airports to be used. Airports closest to ignition center prioritized. | int [-] | 8  [max = X]  [-1 indicates all  available airports] |
| *airports* | Airport agent definitions, using gps coordinates and takeoff/landing types (`runway`, `vertipad`, `water`) | List(dict{  “takeoff\_landing\_types”: TakeoffLandingType,  “gps\_coords” : (lat,lon)}) | [{"takeoff\_landing\_types": "vertipad",  "gps\_coords":  [38.6934,-9.4192]},] |
| *protection\_locations* | The positions of locations (GPS/ Euclidean) agents should prioritize in protecting | List(dict{  “gps\_coords” : (lat,lon)}) | [{"gps\_coords":  [38.760, - 9.44]] |
| *urban\_locations* | The positions of locations (GPS/ Euclidean) for additional urban areas based on elliptical dimensions [m] and angles. | List(dict{  “gps\_coords” : (lat,lon),  “radius”: [x,y],  “angle” : [deg]}) | [{“gps\_coords":  [38.765, -9.424], "radius": [100, 50],  “angle”: 90}] |
| *deploy\_osm\_waters* | Whether OSM obtained water locations and polygons should be used. | bool | true |
| *water\_sources* | The positions of water sources (excluding OSM obtained) in GPS/ Euclidean coordinates. | List(dict{  “gps\_coords” : (lat,lon)}) | [{"pos": [200,40000]}] |
| *response\_time* | The time between ignition start (mission\_start) and the agent’s departure from their bases | int [s] | 3600 |
| *takeoff\_interval* | The time between subsequent departures at airbases | int [s] | 120 |
| *turnaround\_time* | The time required for each aircraft to hold at the airbase between subsequent missions- this time is added to refueling/ recharge times | int [s] | 300 |
| *suppression\_altitude* | Altitude [m] for agent to drop suppressant or water onto fire | float [m] | 50  [0 ≤ *altitude*] |
| *resupply\_altitude* | Altitude [m] for agent to resupply from water source | float [m] | 15  [0 ≤ *altitude*] |
| *enable\_nighttime\_operations* | Specifies whether aerial agents can conduct missions past sunset | bool [true/false] | true |
| *scoop\_time* | The duration an agent needs to pick up water from source | float [s] | 30 |
| *distance\_cost\_weight* | Prioritizes selecting fire fronts closest to water (higher 🡪 closer). | float [-] | 1  [Values are unbounded, recommendation:  0 ≤ weight ≤1] |
| *vip\_cost\_weight* | Prioritizes selecting fire fronts closest to VIP locations (higher🡪closer) | float [-] | 1  [Values are unbounded, recommendation:  0 ≤ weight ≤1] |
| *priority\_cost\_weight* | Prioritizes selecting fire fronts closest to residential areas (higher🡪closer) | float [-] | 1  [Values are unbounded, recommendation:  0 ≤ weight ≤1] |
| *vegetation\_cost\_weight* | Prioritizes selecting fire fronts closest to flammable terrain (higher🡪 closer) used in “vegetation” priority | float [-] | 1  [Values are unbounded, recommendation:  0 ≤ weight ≤1] |
| *topography\_cost\_weight* | Prioritizes selecting fire fronts closest to areas where the terrain slope and wind would increase fire spread (higher🡪closer) | float [-] | 1  [Values are unbounded, recommendation:  0 ≤ weight ≤1] |
| *terrain\_inputs* | Dictionary defining all TerrainParameters | dict [-] | “terrain\_inputs” : {[See Table 2} |
| *agents* | Dictionary defining the agent’s file name, suppression tactic and locations. The locations indicate their starting positions and match the order of the `airports` inputs.  To see how to define the aircraft file, see Table 4.  The suppression tactic must follow that shown in Suppression Tactic Parameters | dict [-] | [{file\_name": "example\_aircraft\_1.json",  "agents\_per\_base": [1,1 1,1 ,1],  "suppression\_tactic": { "main": {  "select\_poi": "indirect",  "track\_poi": "indirect",  "suppress": "indirect"},  "alternative” : “threshold": [17, 4],  "change\_condition": "daytime", "alternative\_tactic": {  "select\_poi": "vip"}}}}] |
| *run\_api\_for\_atmosphere* | Specifies whether the weather data should be based on historical data or a mathematical model (True = historical). The historical data is obtained via an API which stores weather data for a specific region and time. | bool [true/false] | false |
| *atmosphere\_inputs* | Dictionary specifying the atmosphere data model inputs (used when `*run\_api\_for\_atmosphere = false`* | dict [-] | “atmosphere\_inputs” : {[See Table 3]} |

## Terrain Specific Parameters

These inputs are placed within the *terrain\_inputs* of the *WildfireParameters*. Most terrain inputs are fixed to ensure consistency amongst the scenario description of the COLOSSUS project. There are still factors that one can vary to conduct individual analyses or sensitivities.

Table 2. Terrain Parameter Inputs

|  |  |  |  |
| --- | --- | --- | --- |
| ***Parameter*** | **Definition** | **Format** | **Default values** |
| *file\_namespace\_operational* | File name of the operational terrain (larger map) to be used. | *str(Path)* | *"SintraSmall\_128x128\_800"* |
| *file\_namespace* | File name of the main fire grid. | *str(Path)* | *"SintraSmall\_360x260\_10m"* |
| *cell\_size* | Cell size (width and length) of the fire map. **This must match the `file\_namespace`** . | *int [m]* | *10* |
| *priority\_map\_sigma* | Defines the distance strength for generating the area around residential areas. Larger sigma 🡪 larger area around residential that is prioritized | *int [-]* | *30*  *[0 ≤ priority\_map\_sigma ≤ 100]* |
| *height\_scale\_factor* | Scales the elevation of the terrain | *float [-]* | *1*  *[0 < height\_scale\_factor]* |
| *import\_features\_osm* | Whether OSM obtained terrain data (vegetation, urban areas, water, etc.) should be used to construct the fire map. | *bool* | *true* |

## Atmosphere Specific Parameters

These inputs are placed within the *atmosphere\_inputs* of the *WildfireParameters*.These inputs define the mathematical model variables for atmosphere approximations. It is advised to use the API for atmosphere data to maintain a higher accuracy as it is historical data, but the mathematical model can be beneficial for sensitivity analyses and independent studies. For the Grand Challenge, these parameters should not be changed but are worthwhile exploring for bias comparison and can be useful in determining SoS solution effectiveness and robustness (**useful when grading**).

Table 3. Atmosphere Parameter Inputs

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Definition** | **Format** | **Default values** |
| *temperature\_range* | Daily temperature range  (high, low, next day low) in °C | List[°C, °C, °C,] | (10.0, 25.0, 10.0) |
| *temperature\_times* | The times matching the daily high and low temperatures | List[hour, hour] | (5,15) 🡪 05:00 and 15:00 |
| *sun\_times* | Sun rise and sun set times | List[hour, hour] | (6, 20)  [0 ≤ *time* ≤ 24] |
| *time\_of\_max\_solar\_height* | Time corresponding to where Sun is at peak (90°) | float [hour] | 12  [0 ≤ *time* ≤ 24] |
| *humidity\_range* | Minimum and maximum daily relative humidity values | List[%, %] | (20,70)  [0 ≤ *humidity* ≤ 100] |
| *wind\_run* | Total wind run of the day | float [km/d] | 500  [0 ≤ *wind\_run*] |
| *general\_winddirection* | Overall tendency of the wind. 0° indicates wind is coming from North | float [°] (0 = wind from North) | 300  [0 ≤ *direction* ≤ 360] |
| *range\_winddirection* | Expected variance in wind direction from general value throughout the day | float [°] | 30  [0 ≤ *direction* ≤ 360] |
| *update\_frequency* | Specifies how frequently weather values should be changed throughout the simulation | int [s] | 600 |
| *latitude* | Latitude of the ignition map center- used for Atmosphere API only | Lat | 38.755 |
| *longitude* | Longitude of the ignition map center- used for Atmosphere API only | Lon | -9.42 |

## Aircraft Agent Specific Parameters

Unlike terrain and atmosphere parameters, there are many more agent parameters that are variable. Additionally, to allow for connection with aircraft design tools, the agent parameters are defined in a separate file to the main wildfire simulation parameters. The agent definition should be referred to in the wildfire simulation input file accordingly.

Table 4 Wildfire Aircraft Agent Parameter Inputs

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Definition** | **Format** | **Default values** |
| *icon* | The file\_path of the SVG to be used for GUI representation. Found in `*wildfire/static/…*` | str(FilePath) | `evtol.svg` |
| *takeoff\_landing\_type* | The applicable air bases for the aircraft:  *runway* = airport/ conventional base similar  *vertipad* = helipad/ UAM base similar  *water* = seaport/ seaplane base similar | str(TakeOff  LandingType) | *vertipad* |
| *autonomous* | Whether the aircraft can operate without a pilot. Currently this has no impact on wildfire strategies or aircraft performance (HINT for future expansion) | bool | *true* |
| *mtom* | Maximum take-off mass of the aircraft | float [kg] | *1988* |
| *empty\_mass* | Operational empty mass of the aircraft | float [kg] | *1628* |
| *payload* | Defines the maximum payload of the aircraft (affects amount of water/ suppressant carry) | float [kg] | 500 |
| *fflow\_rate* | Average flow rate of suppressant/ water as it leaves aircraft | float [m^3 /s] | 1.2 |
| *can\_scoop* | Whether the aircraft can scoop water or if it the aircraft needs to rely on airbase resupply | bool [True/False] | True |
| *scooping\_distance* | The distance required to scoop suppressant from a water source. Impacts feasible water sources. | float [m] | 6 |
| *span* | Defines the aircraft wing span (maximum y length). Impacts feasible water sources. | float [m] | 6 |
| *propulsion\_input* | Defines the propulsion related parameters of the aircraft system. | dict(PropulsionInput) | “propulsion\_input”:{  xxx : xxx  }  Aircraft propulsion input parameters |
| *profile\_parameters* | Defines the mission profile characteristics of the aircraft (speeds, distances and altitudes) | dict(  ProfileParameters) | “profile\_parameters” : {  xxx : xxxx  }  See |

### Aircraft propulsion input parameters

Aircraft propulsion in the toolkit can be defined in terms of both energy and fuel consuming architectures. To accommodate both, a *PropulsionInput* dictionary is defined. Listed are the various inputs one can use to specify the propulsive parameters.

Table 5 Aircraft Propulsion Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Definition** | **Format** | **Default values** |
| *architecture* | The power system the aircraft uses. Choices are between *electric, conventional*  and *hybrid*.  NOTE if *hybrid* is chosen, the *electric* and *fuel* dictionaries are required as additional inputs where each one is a separate instantiation of *PropulsionInput* dictionary. An example is shown in Figure 2. | str | *electric* |
| *hybridization\_ratio* | The degree of hybridization of the aircraft (power relative to fuel consumption).  NOTE this is only required in *hybrid* architectures | float [-]  Bound [0,1] | N/A  (not shown for *electric* or *conventional*) |
| *total\_propellant* | Defines the total fuel (*conventional*) or energy (*battery*) of the system. | float [kg] fuel/  float [kJ] energy | 492445 |
| *reserve\_propellant* | Defines the reserve fuel (*conventional*) or energy *(battery*) of the system. | float [kg] fuel/  float [kJ] energy | 33253 |
| *propellant\_unit* | The unit used to define the *propellant* parameters. Currently only *kg* and *kJ* are accepted | str | “kJ” |
| *refuelling\_rate* | Defines the refuelling (fuel-based aircraft) rate of the aircraft at airbases. | float [kg/s] fuel | 15.14 |
| *charging\_power* | Defines the charging power (energy-based aircraft) of the aircraft at airbases | float [kJ/s] energy | 359.019 |
| *battery\_swap\_enabled* | Whether or not energy-based aircraft have battery swapping technology (no recharging required) | bool | true |
| *battery\_swap\_time* | Defines the time required for battery swapping. | float [s] | 600 |
| *taxi\_out\_fc*  *taxi\_out\_power* | Defines the required fuel consumption/ power for the aircraft during taxi out. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:32.654,  “2128”:42.612} |
| *taxi\_in\_fc*  *taxi\_in\_power* | Defines the required fuel consumption/ power for the aircraft during taxi in. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:32.654,  “2128”:42.612} |
| *transition\_fc*  *transition\_power* | Defines the required fuel consumption/ power for the aircraft during transition (for VTOLs). For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”: 274.267,  “2128”: 330.716} |
| *retransition\_fc*  *retransition\_power* | Defines the required fuel consumption/ power for the aircraft during re-transition (for VTOLs). For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”: 274.267,  “2128”: 330.716} |
| *takeoff\_fc*  *takeoff\_power* | Defines the required fuel consumption/ power for the aircraft during take-off. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:332.452,  “2128”:433.089} |
| *cruise\_climb\_fc*  *cruise\_climb\_power* | Defines the required fuel consumption/ power for the aircraft during cruise climb. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:350.8,  “2128”:385.032} |
| *cruise\_fc*  *cruise\_power* | Defines the required fuel consumption/ power for the aircraft during cruise. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:268.507,  “2128”:285.914} |
| *cruise\_descent\_fc*  *cruise\_descent\_power* | Defines the required fuel consumption/ power for the aircraft during cruise descent. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:195.196,  “2128”:195.027} |
| *landing\_fc*  *landing\_power* | Defines the required fuel consumption/ power for the aircraft during landing. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:332.452,  “2128”:433.089} |
| *hover\_fc*  *hover\_power* | Defines the required fuel consumption/ power for the aircraft during hover (for VTOL). For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:326.543,  “2128”:426.117} |
| *loiter\_fc*  *loiter\_power* | Defines the required fuel consumption/ power for the aircraft during cruise loiter- used when waiting for landing or for scooping. For fuel, up to 2 mass/power pairs are possible, whereas energy-based aircraft more are allowed. | float  or  dict[int (mass), float(fc/power)] | {“1628”:350.8,  “2128”:385.032} |



Figure 2 Example hybrid propulsion input

### Aircraft profile input parameters

Aircraft profile in the toolkit relates to common mission profile data that is used to define the velocities, accelerations and altitudes of the aircraft. Unlike the propulsion inputs, this is invariant to architecture changes. Listed are the various inputs one can use to specify the profile input parameters.

Table 6 Aircraft Mission Profile Input Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Definition** | **Format** | **Default values** |
| *taxi\_out\_duration* | Defines the time required to perform taxi out | float [s] | 60 |
| *taxi\_in\_duration* | Defines the time required to perform taxi in | float [s] | 60 |
| *transition\_duration* | Defines the time required to perform transition for VTOL aircraft. For non-VTOL aircraft, make this value 0 | float [s] | 45.887 |
| *retransition\_duration* | Defines the time required to perform re-transition for VTOL aircraft. For non-VTOL aircraft, make this value 0 s | float [s] | 45.887 |
| *takeoff\_altitude* | Defines the altitude at which take-off is completed. Transition occurs after if possible, else cruise climb begins. | float [m] | 15.24 |
| *takeoff\_climb\_rate* | Defines the vertical speed of the take-off phase. | float [m/s] | 0.508 |
| *takeoff\_ground\_speed* | Defines the horizontal speed, based on the ground/ surface altitude, of the take-off phase. For VTOL this may be 0 m/s. | float [m/s] | 0 |
| *cruise\_climb\_rate* | Defines the vertical speed of the cruise climb phase. | float [m/s] | 3.556 |
| *cruise\_climb\_ground\_speed* | Defines the horizontal speed, based on the ground/ surface altitude, of the cruise climb phase. | float [m/s] | 51.272 |
| *cruise\_altitude* | Defines the altitude at which cruise climb is completed and the transition to cruise occurs. | float [m] | 457.2 |
| *cruise\_speed* | Defines the horizontal speed of the aircraft during cruise. | float [m/s] | 90 |
| *cruise\_descent\_rate* | Defines the vertical speed of the cruise descent phase. | float [m/s]  Positive values indicate downwards movement. | 3.556 |
| *cruise\_descent\_ground\_speed* | Defines the horizontal speed, based on the ground/ surface altitude, of the cruise descent phase. | float [m/s] | 51.272 |
| *landing\_altitude* | Defines the altitude at which cruise descent is completed. Re-transition occurs after if possible, else landing starts. | float [m] | 15.24 |
| *landing\_descent\_rate* | Defines the vertical speed of the landing phase. | float [m/s]  Positive values indicate downwards movement. | 0.508 |
| *takeoff\_ground\_speed* | Defines the horizontal speed, based on the ground/ surface altitude, of the landing phase. For VTOL this may be 0 m/s. | float [m/s] | 0 |
| *loiter\_speed* | Defines the horizontal speed, based on the ground/surface altitude, of the loiter movement. | float [m/s] | 51.272 |

## Suppression Tactic Parameters

These inputs are placed within the *suppression\_tactic* of the *agents* input. This is the crux of the operational strategy of the firefighting aircraft and offers some modularity. Each suppression tactic is based on the *SuppressionTacticInput* class which is based on 3 inputs which correspond to how the aircraft: 1- selects the areas to suppress, 2- tracks these areas for any changes (useful if the progresses or the area is already suppressed by another agent) and 3- suppresses the point of interest (useful to maximize fire suppression or suppressed area). Additionally, the toolkit offers the ability to dynamically swap tactics based on a *change\_condition* input, where a condition is checked and an alternative tactic is employed to replace the main tactic.

Table 7 Suppression Tactic Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| ***Parameter*** | **Definition** | **Format** | **Default values** |
| *main* | An instance of *SuppressionTacticInput* containing *select\_poi, track\_poi* and *suppress* strings. By default the *select\_poi, track\_poi, suppress* tasks are defined as shown. | *dict(*  *“select\_poi”: str,*  *“track\_poi”: str,*  *“suppress”: str* | *{“select\_poi” : “water”,*  *“track\_poi” : “follow\_firefront”,*  *“suppress” : “direct”}* |
| *alternative* | Details the change condition, threshold for change and alternative tactic input of *SuppressionTacticInput.* By default, the *alternative\_tactic* follows the same defaults as the *main* defaults. | *dict(*  *“change\_condition”:*  *str,*  *“threshold”: [float | float,float]*  *“alternative\_tactic”:*  *SuppressionTactic*  *Input* | *{“change\_condition": “daytime",*  *“threshold": [17, 4], "alternative\_tactic": {*  *"select\_poi": "indirect",*  *"track\_poi": "indirect",*  *"suppress": "indirect" } }* |

### SuppressionTacticInput Options

In order to understand the modularity of the suppression tactics, the different methods and explanations for the tasks of *select\_poi, track\_poi* and *suppress* are detailed.

* ***select\_poi***
  + Options: *water, vip, vegetation, topography, indirect*
  + Definitions:
    - *water:* chooses fire front based on that with the highest spread rate, closest water sources and those in greatest threat to *protection\_locations* and urban areas.

Considers *distance\_cost\_weight,* *vip\_cost\_weight, priority\_cost\_weight*

* + - *vip*: does the same as water but prioritizes areas where `protection\_locations` are defined in the input file by ignoring urban areas.   
      Considers *distance\_cost\_weight, vip\_cost\_weight*
    - *topography*: prioritizes fire fronts where elevation is more likely to accelerate fire spread (based on slope data).   
      Considers *distance\_cost\_weight, vip\_cost\_weight, topography\_cost\_weight*
    - *vegetation*: prioritizes fire fronts where nearby vegetation is of higher combustibility or areas which will increase fire spread.   
      Considers *distance\_cost\_weight, vip\_cost\_weight, vegetation\_cost\_weight*
    - *indirect*: creates an elliptical fire line around the center of the fire (if multiple ignition centers it creates a mean center point). Aircraft then select their suppression locations to fulfill the fire line and do not directly suppress the fire, instead trying to constrain the fire from propogating.  
      Does not consider any weighting inputs.
* ***track\_poi*** 
  + Options: *direct, indirect, follow\_firefront*
  + Definitions:
    - *direct*: the agent simply moves to the original selected point of interest and does not do any recalculation, even if the area is suppressed or burnt.
    - *follow\_firefront*: the agent moves to the original selected point of interest but monitors the fire spread rate, ensuring it selects the region with the greatest fire spread. This ensures that if the fire is extinguished or suppressed a new destination is chosen.
    - *indirect*: the agents move to the fire line created by *select\_poi* and checks if the original chosen area is burnt or extinguished to ensure the fire line is then continued correctly. Note setting *track\_poi* to *indirect* functions better if the *select\_poi* is *indirect* as well as otherwise the fire line itself is not being selected for suppression initially.
* ***suppress***
  + Options: *direct, indirect*
  + Definitions:
    - *direct*: the agent orients its suppression angle to maximize burnt area captured whilst minimizing already suppressed areas.
    - *indirect*: the agent orients its suppression angle to maximize the fulfillment of the fire line. Note setting *suppress* to *indirect* functions better if the *select\_poi* and potentially *track\_poi* is *indirect* as well as otherwise the fire line itself is not being selected for suppression initially.

### *change\_condition* Options

Agents can dynamically swap their tactics from *main* to *alternative* based on the inputted *change\_condition* and its *threshold*. This section will cover the options for these two parameters and their meaning.

* ***change\_condition***
  + Options: *no\_change, runtime, daytime, residential, burnt\_area, distance*
  + Definitions:
    - *no\_change*: the agents do not undergo any tactic swapping and maintain their *main* tactic input. This is the default condition. Note it is possible that when the agents are fulfilling *indirect* attack for their *select\_poi* that the fire line is completed or is detected to fail (this can happen in unique occasions where the fire propagation is very one directional), in such occurrences the agents will automatically swap to the default *SuppressionTacticInput* of *water* suppression.

The *threshold* is not defined / disregarded in this condition.

* + - *runtime*: the agents swap tactics when a certain runtime is met, in hours from simulation start (ie: 4 indicates 4 hours from the mission start time)  
      The *threshold* is defined as an integer of hours from mission start time.
    - *daytime*: the agents swap tactics between two hours of the day, allowing time-based modularity for daytime/ nighttime tactics or time-specific tactics.  
      The *threshold* is defined as a tuple of daytime hours (ie: [10,18.5] indicates that from 10AM to 6:30 PM the agents will swap tactic). Between the hours listed, the agents will swap to their *alternative* tactic.
    - *residential*: the agents swap tactic based on the nearest distance between the fire fronts and urban areas.   
      The *threshold* defines the maximum allowable distance [m] between the fire and the urban bounds that the agents will allow before swapping to their *alternative\_tactic­* (ie: 150 means that the agents will swap tactics only when the fire reaches 150m to the nearest urban bounds.
    - *burnt\_area*: the agents swap tactic when the total fire burnt area exceeds the given threshold.  
      The *threshold* is defined in burnt area [m2] (ie: 10000 means that the tactic is swapped when the total fire burnt area exceeds 10’000 m2/ 1 hectare)
    - *distance*: the agents swap tactic based on the nearest distance between the fire fronts and the fire line. It is possible that the agents suppress the fire, making the distance between the next fire front and the fire line greater than the threshold, resulting in a continuation of the *main* tactic.  
      The *threshold* is defined in the maximum allowable distance [m] between the fire line and the nearest fire front. It is advised to use this condition only if the agents are conducting indirect attack with the fire block to prevent anomalous behavior.