Satellite Toolbox and Mission Control

Version: 2025-05-21

Table of Contents

1 Introduction

`ltesat` is a companion software of Amarisoft eNB/gNB for any NTN scenario and features two main functions:

1.1 Toolbox mode

It can be used 'offline' like a toolbox to replay or plan any satellite orbit. It accepts various inputs for satellite description which are compatible with eNB/gNB NTN configuration and can output the satellite trajectory with several formats: azimuth/elevation from a given ground position, groundtrack or ECEF coordinates.

It can also be used as a pass predictor over a given ground position. It can also export a given satellite description to another format, e.g to generate a TLE file.

1.2 Mission control mode

Secondly, it can be used as a 'mission control' software to control several Amarisoft eNB/gNB working with a worldwide LEO constellation. The configuration file contains a mapping between ground cells and Amarisoft eNB/gNBs with an NTN configuration, along with the definition of a satellite constellation. `ltesat` will configure in realtime the eNB/gNBs to select the serving satellites for its NTN cells, in order to allow a continuous coverage in time. `ltesat` offers the same comprehensive logging mechanism as other Amarisoft products and smoothly interfaces with Amarisoft GUI.

2 Requirements

2.1 Hardware requirements

- `ltesat` is not a performance-intensive executable and can run on any machine.

2.2 Software requirements

- A 64 bit Linux distribution. Fedora 39 is the officially supported distribution. The following distributions are known as compatible:
  - Fedora 22 to 39
  - Cent OS 7
  - Ubuntu 14 to 22

  Your system requires at least GLIBC 2.17.

3 Installation

`ltesat` can be run directly from the directory where it was unpacked.

3.1 Linux setup

`ltesat` does not need external packages.

3.2 Licensing

`ltesat` needs a valid Amarisoft license with the NTN option to run.

3.3 Initial testing

- To display the satellite groundtrack:

  ```
  ./ltesat sat-groundtrack.cfg
  ```

- To act as a mission control:

  ```
  ./ltesat -m sat-mc.cfg
  ```

  In mission control mode, the `ltesat` runs until it receives a `Ctrl+C` termination command.

4 Configuration reference

4.1 Configuration file syntax

The main configuration file uses a syntax very similar to the Javascript Object Notation (JSON) with few extensions.

1. Supported types:
   - Numbers (64 bit floating point). Notation: `13.4`
   - Complex numbers. Notation: `1.2+3*I`
   - Strings. Notation: `"string"`
   - Booleans. Notation: `true` or `false`.
   - Objects. Notation: `{ field1: value1, field2: value2, .... }`
   - Arrays. Notation: `[ value1, value2, .... ]`
2. The basic operations `+`, `-`, `*` and `/` are supported with numbers and complex numbers. `+` also concatenates strings. The operators `!`, `||`, `&&`, `==`, `!=`, `<`, `<=`, `>=`, `>` are supported too.
3. The numbers `0` and `1` are accepted as synonyms for the boolean values `false` and `true`.
4. {} at top level are optional.
5. " for property names are optional, unless the name starts with a number.
6. Properties can be duplicated.
   If properties are duplicated, they will be merged following [JSON merge rules], page 5, with overriding occuring in reading direction (last overrides previous).
   Ex:

```
{
    value: "foo",
    value: "bar",
    sub: {
        value: "foo"
    },
    sub: {
        value: "bar"
    }
}
```

   Will be equivalent to:

```
{
    value: "bar",
    sub: {
        value: "bar"
    }
}
```

7. Files can be included using *include* keyword (must not be quoted) followed by a string (without :) representing the file to include (path is relative to current file) and terminating by a comma.
   Arrays can't be included.
   Merge will be done as for duplicate properties.
   If *file1.cfg* is:

```
    value: "foo",
    include "file2.cfg",
    foo: "foo"
```

And *file2.cfg* is:

```
value: "bar",
foo: "bar"
```

Final config will be:

```
{
    value: "bar",
    foo: "foo"
}
```

8. A C like preprocessor is supported. The following preprocessor commands are available:

   `#define` *var expr*
   : Define a new variable with value *expr*. *expr* must be a valid JSON expression. Note that unlike the standard C preprocessor, *expr* is evaluated by the preprocessor.

   `#undef` *var*
   : Undefine the variable *var*.

   `#include` *expr*
   : Include the file whose filename is the evaluation of the string expression *expr*.

   `#if` *expr* Consider the following text if *expr* is true.

   `#else` Alternative of `#if` block.

   `#elif` Composition of `#else` and `#if`.

   `#endif` End of `#if` block.

   `#ifdef` *var*
   : Shortcut for `#if defined(var)`

   `#ifndef` *var*
   : Shortcut for `#if !defined(var)`

   In the JSON source, every occurrence of a defined preprocessor variable is replaced by its value.

9. Backquote strings: JSON expression can be inserted in backquote delimited strings with the `${expr}` syntax. Example: `` `abc${1+2}d` `` is evaluated as the string `"abc3d"`. Preprocessor variables can be used inside the expression. Backquote strings may span several lines.

4.1.1 JSON merge rules

Merge overriding direction depends on context, i.e source may override destination or the opposite.
JSON merge is recursive for Objects and Arrays.

Example, merging

```
{
    foo: { value: "bar" },
    same: "one",
    one: 1
}
```

with

```
{
    foo: { value: "none", second: true },
```

```
    same: "two",
    two: 1
}
```

Will become:

```
{
    foo: { value: "bar", second: true },
    same: "one",
    one: 1
    two: 1
}
```

assuming first object overrides second one.

In case of Array merging, the final array length will be the maximum length of all merged arrays.
For each element of the final array, merge will be done considering defined elements only.
Ex:

```
{
    array: [0, 1, 2, { foo: "bar" } ],
    array: [3, 4],
    array: [5, 6, 7, { bar: "foo" }, 8 ]
}
```

Will be merged to:

```
{
    array: [5, 6, 7, { foo: "bar", bar: "foo" }, 8 ],
}
```

4.2 Global properties

`constellation_list`
: Optional array of constellation objects. If absent, a single satellite descriptor object is parsed.See [single_sat], page 8. Each constellation object can either be:

  - A shell of circular orbit planes described by the following parameters:

    `n_planes` Integer (range 1 to 72). Number of orbital planes. The orbital planes are spread evenly in longitude of ascending node.

    `n_sats_per_plane`
    : Integer (range 1 to 72). Number of satellites per orbital plane. The satellites are spread evenly in a given plane. The total number of satellites in the shell is given by `n_planes * n_stats_per_plane`.

    `inclination`
    : Float (range 0.0 to 180.0). Inclination in degrees of the orbital planes.

    `altitude` Float (range 200e3 to 3700e3). Altitude in meters of the shell.

  - An array of single satellite descriptor objects.See [single_sat], page 8,
  - A combined TLE file containing several TLE, described by the following parameter:

    `multiple_tle_filename.`
    : String. Filename where the TLEs are stored. The TLEs should be stored in sequence, blank lines are ignored.

`output_mode`
: Optional enumeration: az_el, groundtrack, ecef, pass (default = az_el). Selects the wanted output when operating in 'toolbox' mode:

  `az_el` Displays distance, azimuth, elevation and doppler. Needs at a least a valid ground cell definition. See [single_cell], page 9,

  `groundtrack`
  : Displays latitude, longitude and altitude.

  `ecef` Displays ECEF state vectors for position (X, Y, Z) and velocity (VX, VY, VZ).

  `pass` Displays next pass information. Needs at least a valid ground cell definition. See [single_cell], page 9,

`output_list`
: Optional boolean (default = false). When true, the output will consist of a single point in time, defined by `start_time` for each of the satellites.
When false, the output is a trajectory for a given satellite (as given by `sat_id`) across time (as given by `start_time`, `step_ms` and `n_steps`).

`sat_id` Optional integer (default = 0). When multiple satellites are defined, and if `output_list` is false, selects the satellite of interest.

`step_ms` Optional integer (range 10 to 600000, default 1000). Step duration in milliseconds for the satellite trajectory.

`n_steps` Optional integer (range 1 to 1440, default 300). Number of steps for the satellite trajectory. For the `pass` output mode, it is the consecutive number of passes to compute.

`start_time`
: Optional string, formatted `"YYYY-MM-DDTHH:MM:SS[.mmm]"` (ISO 8601 format) in UTC time. Defines the starting point for the satellite trajectory or pass estimation. If absent, the current time at execution is taken.

`min_elevation`
: Optional integer (range 0 to 80, default 10). Used only in mission control mode or `pass` output mode. Minimum elevation, in degrees, to consider for a valid overhead pass.

`cells_list`
: Optional array of objects. Each object is a single ground cell descriptor object.See [single_cell], page 9,
If absent, a single ground cell descriptor object is parsed directly.

`export_type`
: Optional enumeration: none, json, tle (default = none). Exports the satellites information, timestamped at `export_time` in the file specified by `export_filename`.
If set to `json`, it will export a JSON array of single satellite descriptor objects in the `ephemeris` format, compatible with `ltesat` or Amarisoft LTE gNB/eNB input.
If set to `tle`, it will export the generated TLEs of the satellites, compatible with the `multiple_tle_filename` input of `ltesat` or any other third party software using TLEs.

`export_filename`
: Optional string. Mandatory if `export_type` is not `none`, ignored otherwise.
Filename where the exported data is written.

`export_time`
: Optional string, formatted `"YYYY-MM-DDTHH:MM:SS[.mmm]"` (ISO 8601 format) in UTC time. Ignored if `export_type` is `none`. Epoch to use when generating the exported data. If absent, the current time at execution is taken.

4.3 Single satellite descriptor object

This object is placed either as an element of a constellation object of array type (See [constellation], page 6) or at the top level of the configuration file (for a single satellite configuration).
Its configuration is compatible with the `ntn` object in Amarisoft eNB/gNB configurations.
It contains the following parameters:

`tle_filename`
: Optional string to configure satellite ephemeris from a Two Line Elements (TLE) file.
The file shall contain only the two lines of data and optionally a title line.
When the parameter is present, `ephemeris` is ignored.

`default_ephemeris`
: Optional enumeration: `geo`, `meo`, `leo`. Default is `geo`.
If `ephemeris` is absent, a default satellite ephemeris is generated so that the satellite is overhead the eNB ground position at the time specified by `start_time`. The GEO and MEO satellite will be placed on the equatorial plane (zero inclination) at the longitude of the eNB ground position.
The LEO satellite will be initially placed at the zenith of the eNB position.

`default_altitude`
: Optional float, range 200e3 to 36000e3. If `default_ephemeris` is used and set to `meo` or `leo`, this parameter (in meters) allows to override the altitude of the chosen orbit.
The parameter is ignored otherwise. The default values are 550e3 for `leo` and 8063e3 for `meo`.

`default_elevation_offset`
: Optional float, range -90 to 90, default = 0. If `default_ephemeris` is used and set to `leo`, this parameter (in degrees) allows to adjust the initial elevation of the satellite compared to the zenith position.
The parameter is ignored otherwise. Negative values will place the satellite before its zenith pass and positive values after the zenith.

`ephemeris`
: Optional object to configure satellite ephemeris in the form of orbital parameters.
The ephemeris configuration is understood in a fixed ECI reference frame aligned with the J2000 vernal equinox, like a TLE configuration. If absent and if `tle_filename` and `sv_filename` are also absent, a default ephemeris is generated.
Contains the following parameters:

  `eccentricity`
  : Float value. Range 0 to 0.99. Eccentricity, unitless

  `inclination`
  : Float value. Range 0 to π. Inclination, in radians.

  `semi_major_axis`
  : Float value. Semi-major axis, in meters.

`longitude`
: Float value. Range 0 to 2π. Longitude of the ascending node, in radians.

`periapsis`
: Float value. Range 0 to 2π. Argument of periapsis, in radians.

`anomaly`
: Float value. Range 0 to 2π. Mean anomaly of the satellite on its orbit at `epoch`, in radians.

`epoch`
: String, formatted `"YYYY-MM-DDTHH:MM:SS[.mmm]"` (ISO 8601 format) in UTC time. Epoch for the given orbital parameters.

4.4 Single ground cell descriptor object

This object describes a ground position used as reference for `az_el` and `pass` output. For the mission control mode, it describes a ground footprint of the satellite beams.
It contains the following parameters:

`ground_position`
: Object describing the cell center, contains the following parameters:

  `latitude`
  : Float value. Range -90 to 90. Degrees of latitude.

  `longitude`
  : Float value. Range -180 to 180. Degrees of longitude.

  `altitude`
  : Optional float value (default = 0). Range -1000m to 20km. Altitude in meters.

`radius`
: Optional float (range 1e3 to 15000e3, default 400e3, used only in mission control mode) Radius of the cell in meters.

`supported_constellations`
: Optional array of integer (used only in mission control mode). Allows to restrict only the satellites of the given constellation indices of the `constellation_list` array to cover the cell. See [constellation], page 6,

4.5 Mission control specific parameters

These parameters are used only when the program is launched with the `-m` option. They are also placed at the top level of the configuration file.

`ran_node_list`
: Array of objects to describe the connection to Amarisoft eNB/gNB and the mapping between RAN cells and ground cells. Each object contains the following parameters:

  `addr`
  : String. IP adress and port of the COM port of the eNB/gNB.

  `label`
  : Optional string. Custom label for this eNB used in logs and console output.

  `mapping_list`
  : Array of objects. Describes the mapping between a ground cell and one or more RAN cells. Each ground cell can only be referenced in one RAN node and each RAN cell can only be mapped to one ground cell. Each object contains the following parameters:

    `ground_cell_id`
    : Integer. Index of the ground cell in the `cell_list` array

`ran_cell_list`
: Array of integers. `cell_id` values of the RAN cells mapped to this ground cell. With more than 1 cells, the mission control will ensure a small overlap between satellite pass to offer a continuous coverage of the cell and the possibilities for the UEs to perform handovers.

`pass_overlap`
: Optional integer (range 10 to 120, default 30). Overlap duration in seconds of the coverage of the ground cells with more than one RAN cell.

`dump_ran_status`
: Optional boolean. Print useful information in the console output. This options is preferred when no GUI is connected to the `ltesat` tool.

`log_filename`
: String. Set the log filename. If no leading `/`, it is relative to the configuration file path.

`log_options`
: String. Set the logging options as a comma separated list of assignments.

  - *layer*.level=*verbosity*. For each layer, the log verbosity can be set to `none`, `error`, `info` or `debug`. In debug level, the content of the transmitted data is logged.
  - *layer*.max_size=*n*. When dumping data content, at most `n` bytes are shown in hexa. For ASN.1, NAS or Diameter content, show the full content of the message if `n > 0`.
  - time=[sec|short|full]. Display the time as seconds, time only or full date and time (default = time only).
  - time.us=[0|1]. Dump time with microseconds precision.
  - file=cut. Close current file log and open a new one.
  - file.rotate=now. Rename current log with timestamp and open new one.
  - file.rotate=*size*. Rename current log every time it reaches *size* bytes open new one. Size is an integer and can be followed by K, M or G.
  - file.path=*path*. When log rotation is enabled, move current log to this path instead of initial log path.
  - append=[0|1]. (default=0). If 0, truncate the log file when opening it. Otherwise, append to it.

  Available layers are: PROD

`log_sync`
: Optional boolean (default = false). If true, logs will be synchronously dumped to file.
  Warning, this may lead to performances decrease.

`com_addr`
: String. IP adress and port on which the `ltesat` software can receive websocket connection

5 Examples

5.1 Toolbox usage

Fetch a well-known TLE from online source in a file, e.g. here is the ISS(ZARYA) TLE from 2024/03/07, stored in `iss.tle`:

```
1 25544U 98067A   24067.37790297  .00014521  00000-0  26457-3 0  9993
2 25544  51.6412  99.9639 0005680 335.2557 148.0116 15.49704815442763
```

5.1.1 Overhead pass

Here is the configuration file to get the next 10 overhead passes over Paris, with a minimum elevation of 20, from the 2024/03/10 onward, stored in `sat-pass.cfg`:

```
/* ltesat configuration file version ##VERSION##
 * Copyright (C) 2024-2025 Amarisoft
 */
{
    output_mode: "pass",
    min_elevation: 20,
    n_steps: 10,

    start_time: "2024-03-10T00:00:00",

    ground_position: {
        latitude: 48.8538,
        longitude: 2.3475
    },
    tle_filename: "iss.tle"
}
```

It will give the following output:

```
[user@host]$ ./ltesat sat-pass.cfg
Satellite utility version 2024-03-15, Copyright (C) 2012-2025 Amarisoft
[OK] TLE at epoch 2024-03-07T09:04:11.000
               T_start(UTC)      T_max(UTC) Elevation(d)      T_end(UTC)
    2024-03-10T00:20:18.740    00:22:27.640       50.276    00:24:36.540
    2024-03-10T01:57:11.100    01:59:26.580       81.349    02:01:42.060
    2024-03-10T21:55:07.940    21:57:20.320       65.740    21:59:32.710
    2024-03-10T23:32:01.760    23:34:11.600       52.915    23:36:21.450
    2024-03-11T01:08:59.030    01:11:12.550       65.108    01:13:26.080
    2024-03-11T02:45:58.970    02:47:52.220       37.528    02:49:45.480
    2024-03-11T21:07:09.920    21:09:10.410       44.918    21:11:10.910
    2024-03-11T22:43:42.410    22:45:54.320       59.901    22:48:06.240
    2024-03-12T00:20:44.720    00:22:55.620       55.556    00:25:06.530
    2024-03-12T01:57:35.880    01:59:43.090       55.364    02:01:50.310
```

5.1.2 Trajectory

Here is the configuration file to get the detailed trajectory for one of those pass, with one point every 10 seconds over 5 minutes, stored in `sat-traj.cfg`:

```
/* ltesat configuration file version ##VERSION##
 * Copyright (C) 2024-2025 Amarisoft
 */
{
    output_mode: "az_el",
    n_steps: 30,
    step_ms: 10000,
    start_time: "2024-03-10T01:57:00",

    ground_position: {
        latitude: 48.8538,
        longitude: 2.3475
    },
    tle_filename: "iss.tle"
}
```

It will give the following output:

```
[user@host]$ ./ltesat sat-traj.cfg
Satellite utility version 2024-03-15, Copyright (C) 2012-2025 Amarisoft
[OK] TLE at epoch 2024-03-07T09:04:11.000
                Time(UTC)  Distance(km)  Azimuth(d)  Elevation(d)  Doppler(ppm)
   2024-03-10T01:57:00.000      1126.994     291.677        17.929        21.805
   2024-03-10T01:57:10.000      1061.955     291.937        19.706        21.571
   2024-03-10T01:57:20.000       997.688     292.233        21.673        21.288
   2024-03-10T01:57:30.000       934.361     292.574        23.865        20.942
   2024-03-10T01:57:40.000       872.186     292.971        26.328        20.516
   2024-03-10T01:57:50.000       811.436     293.444        29.115        19.987
   2024-03-10T01:58:00.000       752.465     294.017        32.295        19.324
   2024-03-10T01:58:10.000       695.733     294.731        35.949        18.486
   2024-03-10T01:58:20.000       641.842     295.646        40.175        17.419
   2024-03-10T01:58:30.000       591.574     296.869        45.082        16.055
   2024-03-10T01:58:40.000       545.940     298.594        50.778        14.314
   2024-03-10T01:58:50.000       506.200     301.214        57.346        12.110
   2024-03-10T01:59:00.000       473.844     305.672        64.770         9.381
   2024-03-10T01:59:10.000       450.471     314.804        72.789         6.126
   2024-03-10T01:59:20.000       437.526     340.282        80.217         2.454
   2024-03-10T01:59:30.000       435.940      44.006        81.573        -1.404
   2024-03-10T01:59:40.000       445.835      79.137        74.933        -5.155
   2024-03-10T01:59:50.000       466.479      90.735        66.872        -8.541
   2024-03-10T02:00:00.000       496.533      96.016        59.239       -11.418
   2024-03-10T02:00:10.000       534.407      98.996        52.428       -13.761
   2024-03-10T02:00:20.000       578.563     100.908        46.498       -15.621
   2024-03-10T02:00:30.000       627.674     102.240        41.385       -17.080
   2024-03-10T02:00:40.000       680.662     103.224        36.984       -18.220
   2024-03-10T02:00:50.000       736.686     103.983        33.183       -19.115
   2024-03-10T02:01:00.000       795.098     104.588        29.882       -19.822
   2024-03-10T02:01:10.000       855.404     105.083        26.994       -20.385
   2024-03-10T02:01:20.000       917.223     105.498        24.448       -20.837
   2024-03-10T02:01:30.000       980.263     105.851        22.186       -21.203
   2024-03-10T02:01:40.000      1044.297     106.156        20.162       -21.502
   2024-03-10T02:01:50.000      1109.145     106.423        18.336       -21.748
```

5.1.3 Export

For testing purpose, it could be useful to generate a 'virtual TLE' that simulates a satellite passing overhead a given location at a given time. Here is an example with the `default_ephemeris` and `export_type` parameter, stored in `sat-export.cfg`:

```
/* ltesat configuration file version ##VERSION##
 * Copyright (C) 2024-2025 Amarisoft
 */

{
    output_mode: "az_el",
    n_steps: 10,
    step_ms: 30000,
    start_time: "2024-03-15T14:30:00",

    ground_position: {
        latitude: 48.8538,
        longitude: 2.3475
    },

    default_ephemeris: "leo",
    default_altitude: 700e3,
    default_elevation_offset: -60,

    export_type: "tle",
    export_time: "2024-03-15T00:00:00.000",
    export_filename: "pass_14h30.tle"
}
```

And the resulting TLE `pass_14h30.tle`:

```
1 99999U 24999A   24075.00000000 -.00000000  00000-0  00000-0 0    11
2 99999  48.8538 305.9746       0 357.5651 150.3878 14.57888518000073
```

5.2 Mission control

Mission control file are too long to be included as is in this documentation, but a sample file is given in `sat-mc.cfg`