2.9 Consistency check

The consistency check function is run automatically before the start of an event in order to detect erroneous or inconsistent values in the scenario definition (see Figure 32). You can also trigger it manually as illustrated in Figure 33. The aim of the consistency check is to prevent runtime exceptions during the simulation due to incorrect input values.

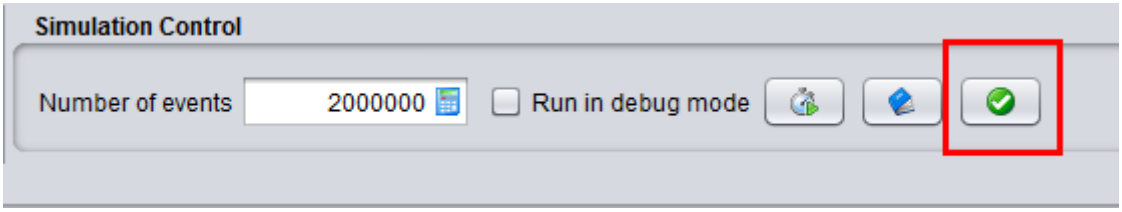

Figure 33: Consistency check button in the scenario tab

The consistency check includes:

- Warnings if odd configuration are detected for random parameters, such as negative antenna height, etc;
- Consistency between inter-dependent parameters such as: C/I, C/(N+I), (N+I)/N, and I/N;
- Verification of the application range of propagation models.

In case it fails, the scenario consistency check will pop up a message window like the below

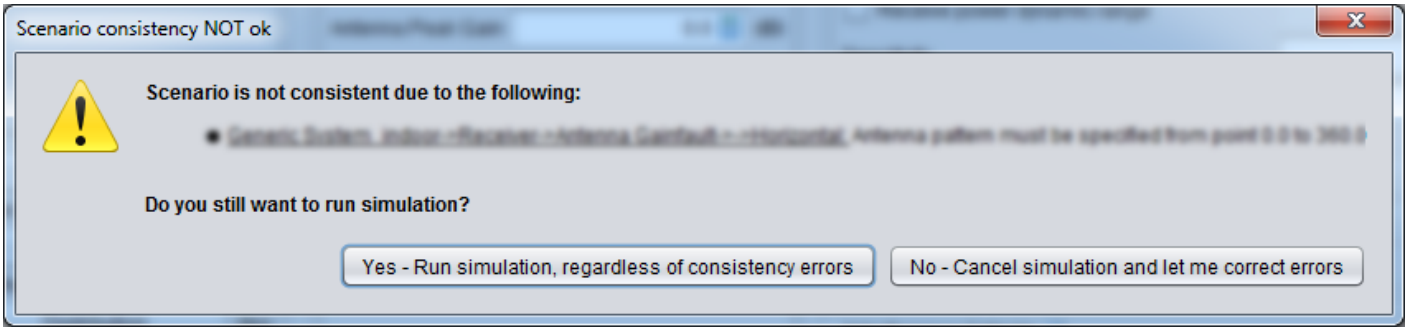

Figure 34: Scenario consistency check message

These warnings may be ignored in some cases, but you should be aware of the following:

- Consistency checks are generally connected to non valid values which cause runtime exceptions during the simulation
- Each runtime exception generates an appropriate error message which is recorded on the seamcat.log file (located on your SEAMCAT home directory):

```
(AntennaGainConfiguration.java:44[evaluate]) - Error evaluating plugin
org.seamcat.model.functions.FunctionException: Specified value (313.7921785343926) is outside bounds [0.0 to 270.0]
    at org.seamcat.function.DiscreteFunction.evaluate(DiscreteFunction.java:90)
    at org.seamcat.model.antenna.HorizontalVerticalAntenna.evaluate(HorizontalVerticalAntenna.java:43)
    at org.seamcat.model.antenna.HorizontalVerticalAntenna.evaluate(HorizontalVerticalAntenna.java:18)
    at org.seamcat.plugin.AntennaGainConfiguration.evaluate(AntennaGainConfiguration.java:42)
    at org.seamcat.simulation.generic.GenericVictimSystemSimulation.collect(GenericVictimSystemSimulation.java:182)
    at org.seamcat.model.engines.InterferenceSimulationEngine.single(InterferenceSimulationEngine.java:145)
    at org.seamcat.model.engines.InterferenceSimulationEngine$1.call(InterferenceSimulationEngine.java:75)
    at org.seamcat.model.engines.InterferenceSimulationEngine$1.call(InterferenceSimulationEngine.java:72)
    at java.util.concurrent.FutureTask.run(Unknown Source)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
    at java.lang.Thread.run(Unknown Source)
```

Note that the yellow marked header format can be changed as described in Annex 20.

- Logging the error messages takes time, hence it significantly increases the simulation time;
- Depending on the scenario (in particular the number of events simulated) the amount of the seamcat.log file might readily grow up to some tens of Mega Byte per simulation run.

It is therefore highly recommended to solve the reported issues prior to run the simulation. Further information on the meaning of consistency check and SEAMCAT error message is available in ANNEX 19:.

Section 14.3.2 presents the implementation of the consistency check in the source code and for plugin development and its graphical results.

Revision #1
Created 2026-04-14 08:54:59 UTC by ECO TECH
Updated 2026-04-14 08:56:14 UTC by ECO TECH