USER MANUAL

Federated Vehicle Testbed Project - User Instruction Guide

Written By

Jacob Carroll
Madeleine Cochrane
Matthew Rimkevicus
Jed Sharman
Michael Shim
Aslan Chadwick
Ryan Black
Timothy Quill
Simon Daniel

Swinburne University of Technology

Contents

1 Introduction **2**
- 1.1 Purpose of Document 2
- 1.2 Project Overview 2

2 Menus System **3**
- 2.1 Main Menu 6
- 2.2 Scenario Config Menu 6
- 2.3 Data Menu 7
- 2.4 Agent Menu 8
- 2.5 Pause Menu 10

3 Configuration **11**
- 3.1 Overview 11
- 3.2 Composition 11
  - 3.2.1 Scenario 11
  - 3.2.2 Agent 11
  - 3.2.3 Communication 11
- 3.3 Using Configurations 12
  - 3.3.1 In the Program 12
  - 3.3.2 With XML Files 12

Chapter 1

Introduction

1.1 Purpose of Document

This document serves as a user manual for the Federated Autonomous Vehicle Testbed Project. It contains information about menu layouts and controls and provides further information on use of the simulation engine. This report is targeted at end-users i.e. people interacting with a running simulation. The information here is primarily focused on knowledge needed to use menus and configuration settings within the environment.

For users wanting information pertaining to extension of the product and configuration of custom components, please consult the Technical Manual.

1.2 Project Overview

The aim of the overall project was to design a simulation environment to be used for the development, testing and demonstration of algorithms for federated fleets of vehicles. The engine in its current form serves as a modular and extensible framework upon which users can create custom agents, maps and scenarios. Users can configure agents with countermeasures and gadgets to meet their needs.

When running a scenario, each agent can be controlled by an AI or a player and can communicate with other friendly agents to coordinate actions. Every action and decision is recorded by the inbuilt data and event recorder for which users can create and select custom data streams.

Chapter 2

Menus System

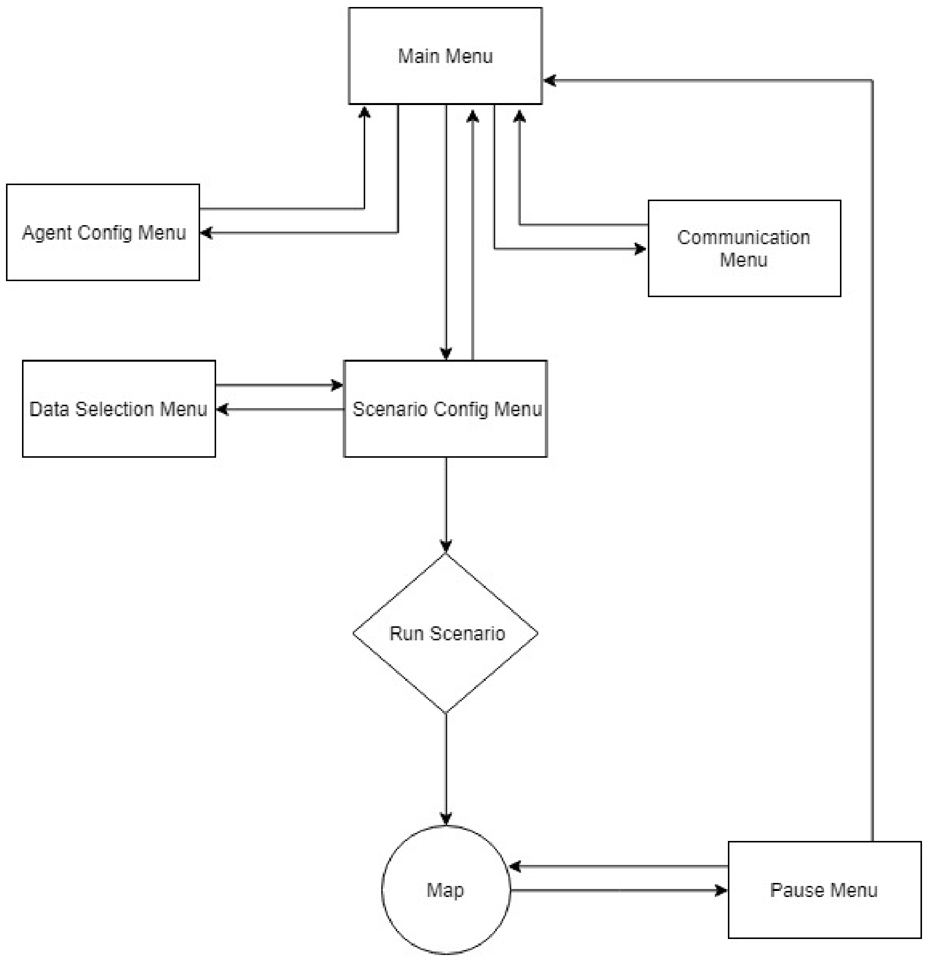


Figure 2.1: Menu Flow

The menus system connects five setup menus and a pause menu allowing users to navigate between the entry point (the main menu) and the initiation/maintanance of a simulation. The flow of this menu structure can be seen in Menu Figure 1. with rectangles representing menus (widgets), diamonds representing actions and ovals representing maps.

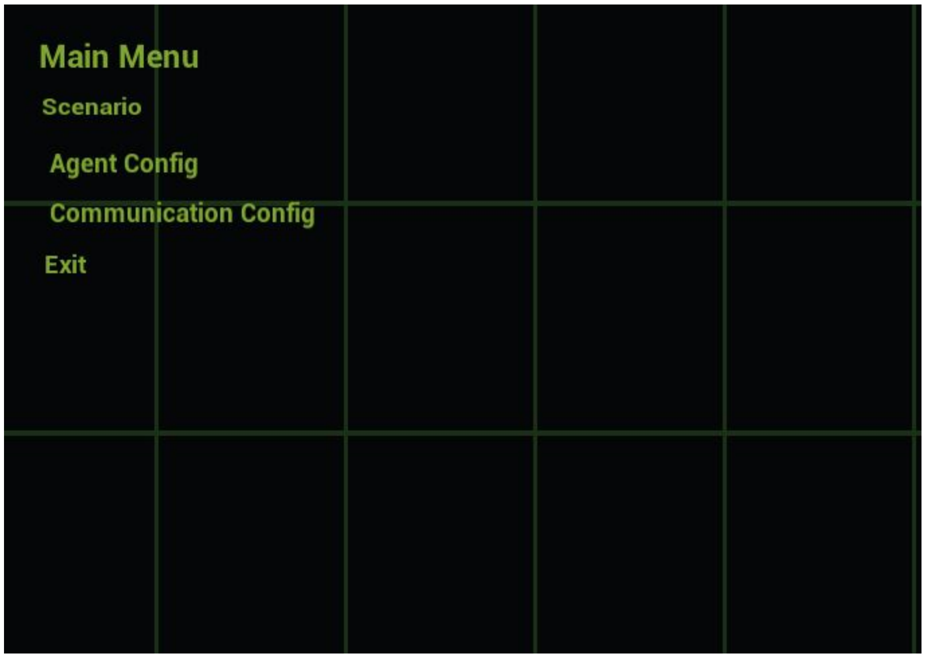
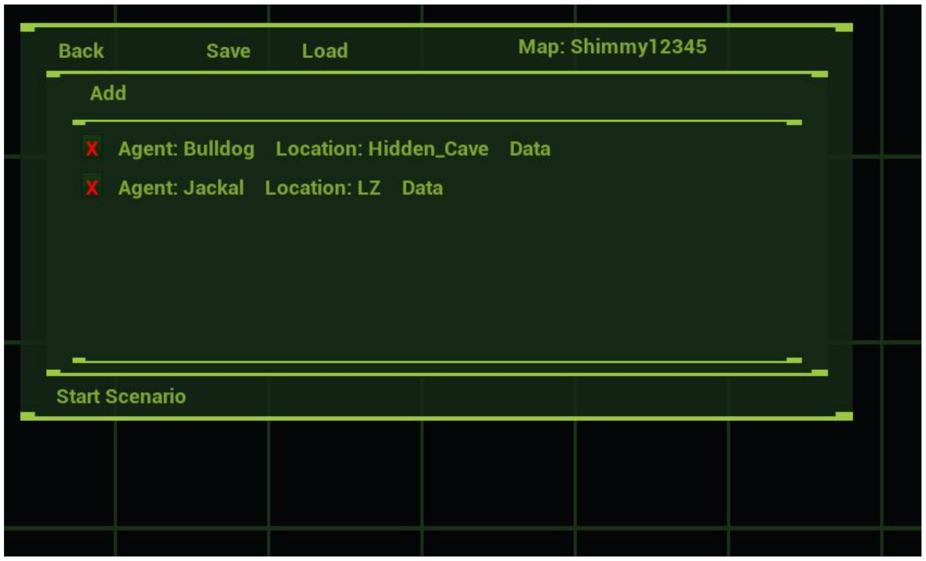


Figure 2.2: Main Menu

2.1 Main Menu

The main menu is the entry point of the menu flow, it can be opened by playing any map using the MainMenuHUD. This is the default HUD for OpenerMap and so acts as an easy way to open up the menu flow for a user. The main menu has four options, Scenario, Agent Config, Communication Config and Exit.

The first three options will open up the corresponding menus and allowing the user to edit the scenario settings, specific agent configuration settings and communication settings respectively. The last option "Exit" will immediately terminate the program and acts as the exit point for the program.



Map: Shimmy12345

Agent: Bulldog Location: Hidden_Cave

Agent: Jackal Location: LZ

Figure 2.3: Scenario Menu

2.2 Scenario Config Menu

The scenario config menu has five points of interest:

1. Back button, top left of screen. This button is used to return the the previous menu.

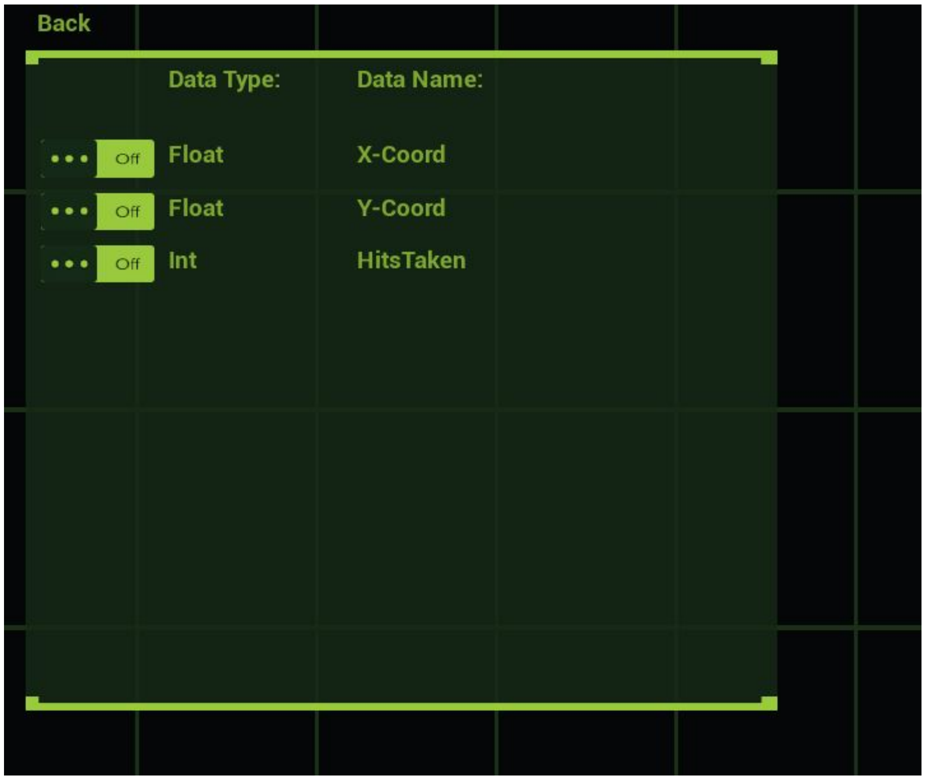
2. Map selection button, top right of screen. This button opens a drop down menu and allows users to select the map to be used in the scenario

3. Add button, below the Back button. This button adds a new blank spawn configuration.

4. Spawn configuration, below the Add button. These configurations allow the user to choose the agent to spawn and the location where the agent

should be spawned. The included data button allows the user to open another menu and select what data to record from the agent once it's spawned.

5. Start button, bottom left of the screen. This button allows the user to run the scenario as configured.



	Data Type:	Data Name:
Off	Float	X-Coord
Off	Float	Y-Coord
Off	Int	HitsTaken

Figure 2.4: Data Menu

2.3 Data Menu

The data menu allows users to toggle on and off the data to be recorded for a given agent spawn. To toggle a data stream click the slider to the left of the streams data type.

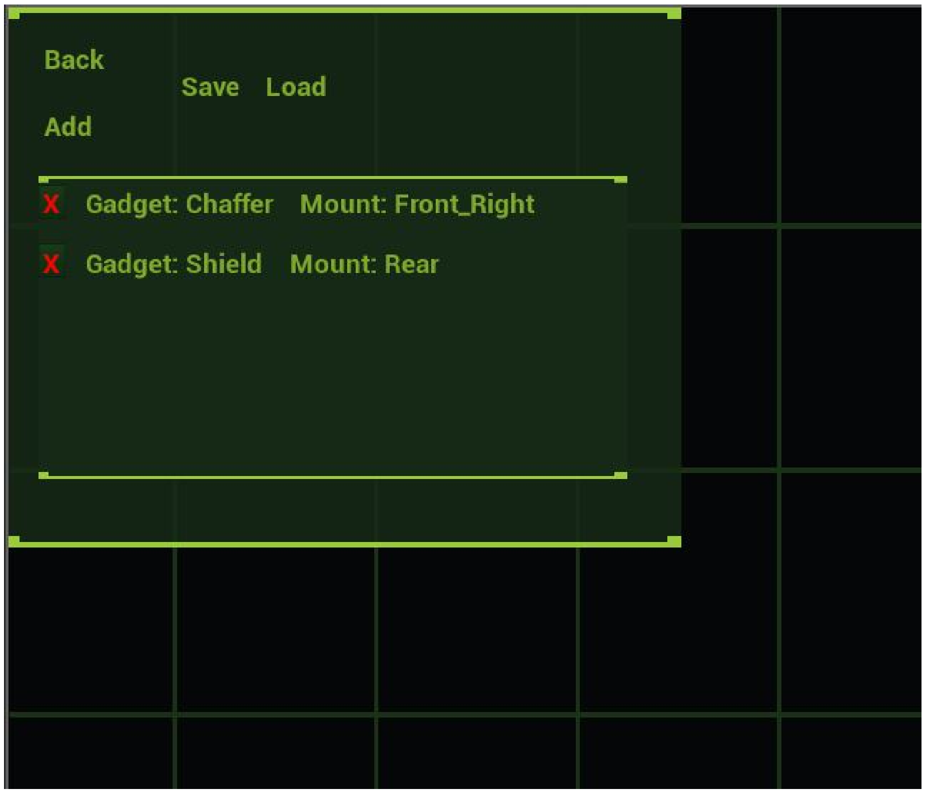
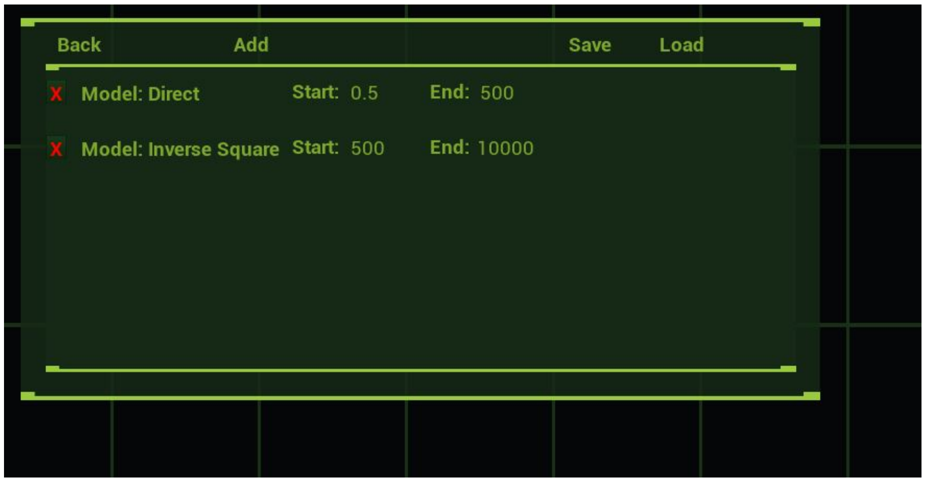


Figure 2.5: Agent Menu

2.4 Agent Menu

The agent menu allows the user to configure an agent by adding and removing gadgets from mounting spots on existing agent blueprints. To add a gadget press the add button, select the gadget to mount and the mounting location.



Model: Direct Start: 0.5 End: 500

Model: Inverse Square Start: 500 End: 10000

Figure 2.6: Communication Menu

5. Communication Menu The communication menu allows users to add and remove frequency ranges and the models which should be used to propogate messages inside a scenario. Ranges can be added and removed using the core-sponding buttons, the range values are in hertz.

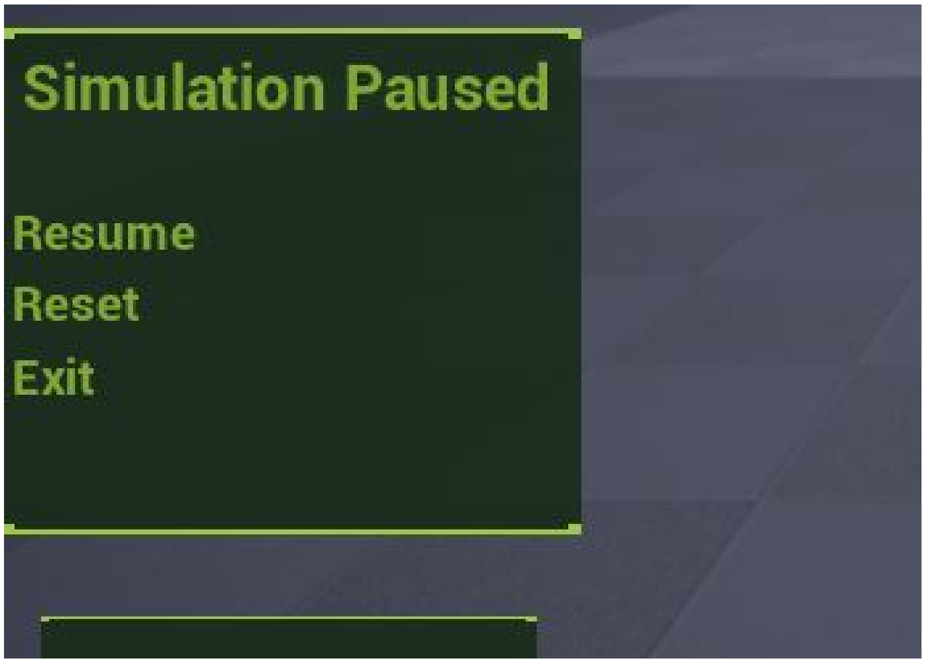


Figure 2.7: Pause Menu.

2.5 Pause Menu

The pause menu is used to navigate in game functionality for the user, it can opened by pressing the pause key (default p) while using the TestbedPlayer-Controller.

Chapter 3

Configuration

3.1 Overview

The configuration system is designed to allow users of the simulation to define and run their own scenarios. The system saves these scenarios, and all relevant components, to file in an xml format. These files contain all the information necessary to recreate a scenario in a human readable format.

3.2 Composition

Scenario configurations rely on a few other configurations. Namely, agents and communications. In order to create a scenario configuration that is more than an empty map you will need to create at least one configuration of each of the aforementioned types.

3.2.1 Scenario

Scenario configurations are the overarching configuration for the scenario. They determine which agent configurations to use and where to spawn them in the map, and what communications configuration to use. They also determine where to save the output files of the data recording systems.

3.2.2 Agent

Agent configurations determine the properties of an agent. What type of agent it is (e.g. Jackal), what countermeasures it is equipped with, and what name it should be represented with in the simulation.

3.2.3 Communication

Communication configurations determine the radio propagation properties to use in the simulation. See the communications documentation for more details.

The configuration determines what propagation model to use by default and what other models to use within ranges of frequencies.

3.3 Using Configurations

Configurations can be stored anywhere in storage, but keep in mind that scenario configurations point to agent and communication configurations by file location so take care when moving files.

3.3.1 In the Program

Scenario

Before jumping into the simulation, the program will first ask you to select or create a new scenario configuration to use. You will need to add a communication configuration to use, and can add agent configurations to use as well. There is always an option to create a new communication or agent configuration. You can set which starting points to use and which agent configurations to use at those starting points. You can use the same agent configuration for multiple starting points, the simulation will append a number to the name of all subsequent agents using the same configuration. E.g. Agent, Agent2, Agent3, etc.

Agent

The program uses the reflection system of the Unreal Engine to find all available options for what class of agent and what countermeasures are available. The name of the agent can be anything, it is merely a way to identify it within the simulation and in data output files.

Communication

The program again uses the reflection system to find all propagation models available for use. These models have been denoted as 'SNR models' because their purpose is to calculate the signal-to-noise ratio between two points. You will need to set which model to use, and can choose to add other models to use within a range of frequencies instead of the default model.

3.3.2 With XML Files

Scenarios can be created solely in XML however you will not have Unreal's reflection system available to help select the correct name of classes you wish to use. The names of blueprint classes in particular will not be what one might expect so it is recommended to use the program to create a configuration using a class to find its name from the subsequently created xml document.

Scenario

Scenario configurations have the following structure:

- Scenario Tag
- Map Tag
- Agents Tag
  - Agent Tag
- Spawns Tag
  - Spawn Tag
    - Name Tag
    - Agent Tag
- Data Recording Output Tag
- Event Recording Output Tag
- Communication Tag

The program can interpret these major tags in any order, but will not look for any sub tags anywhere other than within their major tag.

Scenario Tag The scenario tag is used to identify this configuration as a scenario configuration. Do not remove or change this tag or the program will fail to load this scenario.

Map Tag This tag specifies the name of the map to use, this tag must have a value.

Agents Tag The agents tag houses a list of tags denoting the agent configuration files used by this scenario. It is not necessarily required for a scenario to use agents but the scenario will be empty without them. An agent must be listed here if it is referenced in the spawns tag.

Agent Tag This tag specifies the file name of the agent configuration to use.

Spawns Tag The spawns tag houses a list of spawn tags which have the name of the spawn point and the configuration file of the agent to spawn. Like agents, it is not necessarily required that a scenario have any agents and therefore any spawn points.

Spawns- Spawn Tag This tag contains the name and agents tags of a spawn point.

Spawns- Spawn- Name Tag This tag specifies the name of the spawn point to start an agent at. The names are defined in the program so it may be useful to open the program to see what the names are.

Spawns- Spawn- Agent Tag This tag specifies the agent configuration file to use for this spawn point.

Data Recording Output Tag This tag specifies the output folder of the data recording system. It is recommended that the folder path does not have an ending slash but the system will handle the case that it does.

Event Recording Output Tag This tag specifies the output folder of the event recording system. Like before it is recommended that this value does not have an ending slash.

Communication Tag This tag specifies which communication configuration to use by its file name. This tag must have a value.

Agent

Agent configurations have the following information:

- Agent Tag
- Class Tag
- Name Tag
- Possess Tag
- Gadgets Tag
  - Gadget Tag
    - Name Tag

Agent Tag This tag is used to identify the configuration as an agent configuration.

Class Tag This tag specifies the name of the class this agent should be. It must be a perfect match of the name of an agent class in the program.

Name Tag This tag specifies the name of agents created by this configuration. An agent is not required to have a name but it will reduce the readability of any data recording files.

Possess Tag This tag specifies whether agents of this configuration should be controlled by the player when the simulation starts. If more than one agent of this configuration is created the player will be in control of the last one created.

Gadgets Tag This tag specifies what gadgets are attached to the agent. Gadgets include countermeasures. This tag contains a list of gadget tags which in turn contain a name tag which denotes the name of the class of the gadget.

Gadgets- Gadget Tag This tag describes everything needed to create a gadget in the simulation.

Gadgets- Gadget- Name Tag This tag specifies the name of the class of this gadget. The name must match a gadget class in the program.

Communication

Communication configurations have the following information:

- Communication Tag
- Default Model Tag
- Frequency Ranges Tag
  - Frequency Range Tag
    - SNR Model Tag
    - Min Frequency Tag
    - Max Frequency Tag

Communication Tag This tag is used to identify this configuration as a communication configuration.

Default Model Tag This tag specifies the name of the propagation model to use. The name must match a class in the program.

Frequency Ranges Tag This tag hosts a list of frequency range tags.

Frequency Ranges- Frequency Range Tag This tag specifies an extra propagation model to use within a range of frequencies.

Frequency Ranges- Frequency Range- SNRModel Tag This tag specifies the name of the propagation model to use within this range of frequencies. The name must match a class within the program.

Frequency Ranges- Frequency Range- Min Frequency Tag This tag specifies the minimum value of this range. Inclusive.

Frequency Ranges- Frequency Range- Max Frequency Tag This tag specifies the maximum value of this range. Inclusive.