SignalServer User Manual

Program Version 3.20 Beta

July 18, 2019

Radio propagation simulator by Alex Farrant QCVS, 2E0TDW
Based upon SPLAT! by John Magliacane, KD2BD
Some feature enhancements/additions by Aaron A. Collins, N9OZB
User Manual written by Aaron A. Collins, N9OZB

SignalServer Description:

SignalServer is a multi-threaded RF propagation simulator based upon SPLAT! by Alex Farrant QCVS, 2E0TDW.

SPLAT! Project started in 1997 by John A. Magliacane, KD2BD

This application will generate RF coverage predictions, producing either 2D profile plots (Point-to-Point) or 360 degree polar plots in WGS-84 projection as PPM Bitmaps.

For detailed information and historical reference data related to this project see the SPLAT! documentation. Propagation models added to this project have been sourced from reputable academic sources and all efforts have been taken to ensure their accurate implementation. Not all models are ITU ratified and you use them entirely at your own risk.

Signal Server is a very resource intensive multicore application. Only publish it for common use if you know what you are doing and you are advised to wrap it with another script to perform input validation.

Additional programs/scripts may be required to prepare inputs such as .sdf terrain tiles (srtm2sdf.c), 3D antenna patterns (.ant) and user defined clutter (.udt), or manipulate the bitmap output (.ppm). Some of these are provided in the ***./utils*** directory. More information can be found in the SPLAT! Project documentation.

WARNING: The accuracy of the output is directly proportional to the accuracy of the inputs and the time taken defining and validating them. GIGO!

File extensions and types used by SignalServer:

.asc LIDAR topo data file in ASCII grid format.
.jpg LIDAR topo data file in JPEG format.
.sdf SPLAT! topo data file in SPLAT! format, lo-res 90m (from SRTM3).
.sdf.gz topo data file in SPLAT! data format, gzip compressed, lo-res 90m.
.sdf.bz2 topo data file in SPLAT! data format, bzip2 compressed, lo-res 90m.
-hd.sdf SPLAT! topo data file in SPLAT! format, hi-res 30m (from SRTM1).
-hd.sdf.gz topo data file in SPLAT! data format, gzip compressed, hi-res 30m.
-hd.sdf.bz2 topo data file in SPLAT! data format, bzip2 compressed, hi-res 30m.
.scf signal level color palette file.
.lcf loss level color palette file.
.dcf dbm level color palette file.
.az SPLAT! antenna pattern azimuth data file.
.el SPLAT! antenna pattern elevation data file.
.lrp LIDAR antenna pattern data file.
.udt user defined terrain point clutter data CSV text file.
.sh miscellaneous shell scripts and utility batch files.
.py miscellaneous Python scripts and utility batch files.
.ppm portable pixmap - output plot graphic rendering file (native).
.png portable network graphics - output plot graphic rendering file (converted).
.kml Google Earth Keyhole Markup Language - output file viewable with Google Earth.
.kmz Google Earth Keyhole Markup Language - .kml type file, but zip compressed.

SignalServer Requirements:

* Linux
* GCC,G++
* Multicore CPU (optional)
* ~2GB Memory
* C/C++ system libraries (see below)
* Optional Reference data - SRTM Terrain or ASCII Grid tile(s), Antenna (ANT) pattern files, etc. (see below).

Installing Support Libraries and Packages:

Install any pkgs needed by Signal-Server if necessary. They are very common, basic support libraries, and most Linux distributions come with these already installed. To check it they are installed, one at a time, enter `dpkg -l pkgname*` (add an * after the package name to find all the variations of the base package name). To install them if needed, use `apt install pkgname`. The required packages are:

cifs-utils
net-tools
libbz2
libbz2-dev
libz1
libz-dev

Compiling SignalServer:

If you have the git package installed, you can download SignalServer with the command:
`git clone https://github.com/N9OZB/Signal-Server.git`

You can also use the wget command to download it as well:
`wget https://github.com/N9OZB/Signal-Server.git`

Either command will download the project to the ***./Signal-Server*** directory. Change into it with `cd Signal-Server`.

Finally, change to the ***./src*** directory, and type `make` to compile SignalServer.

Installing SignalServer:

In the ***./src*** directory, type `make install` to install SignalServer into the root directory of the SignalServer project. This is the directory below ***./src*** (**..**). To install to a different directory, you can edit the Makefile and change the **`INSTALL_PATH`** variable to a different directory to install the binaries to, such as /***usr/local/bin*** **or /*opt*/Signal-Server.**

Testing SignalServer:

Run the test suite to provide a confidence test and functional checkout of the package after compilation. To execute, run `./test.sh` in the root directory of the SignalServer package installation. This will create the ***./tests*** directory, and leave the resulting output files of the various test suite plots in that directory.

Installing Reference Data:

Signal server is designed for most of the environments and climates on Planet Earth but Polar region support is limited above extreme latitudes. (Svalbard is ok).

It can run with or without terrain data and can even be used to simulate radiation of other EM emissions like light.

You can install terrain files (.sdf) files into a directory and point Signal-Server to at runtime with the `-sdf` parameter followed by the path where you have installed the files. This is also the case for the other optional data files that SignalServer uses, such as antenna files (`-ant`), color files (`-color`), LIDAR terrain data (`-lid`), 17-class MODIS Landcover files (`-clt`) and user-defined point clutter files (`-udt`).

There are utilities in the ./utils directory to help with converting some of the Reference Data files from other available resource formats. There are utilities to:

Convert Space Shuttle RADAR Topography Mission (SRTM) topo data into the .sdf (SPLAT! Data format) that SignalServer requires. There is a standard resolution utility for converting 90m (SRTM3) data files, and an “HD” version that converts 30m (SRTM1) data files.

Convert common .ant antenna pattern files into the .az and .el (SPLAT! Data format) files used by SignalServer. There is also some information on converting GeoTIFF and LIDAR topo data files there.

-sdf SPLAT! Topographic Elevation Data Files

This option is followed by the directory containing Digital Elevation Models (DEM) SDF formatted tiles. These can be created by converting SRTM tiles (30m or 90m) in HGT format with the [srtm2sdf.c] utility from SPLAT!. At the time of writing, these tiles can be obtained for free from the [USGS website](https://dds.cr.usgs.gov/srtm/).

Note these can be compressed using gzip or bzip2 if desired to save disk space and speed up loading the files. For hi-res (HD) SRTM1 (30m) data, bzip2 compresses the best, and for lo-res SRTM3 (90m) data, gzip seems to achieve the best compression. Either method will work fine for whichever data format and resolution is used.

-lid WGS84 ASCII grid tile (LIDAR) Topographic Elevation Data Files

This option is followed by the path and filename of an .asc file for SignalSever to use. LIDAR files with dimensions and resolution defined in the header. LIDAR data can be used providing it is in ASCII grid format with WGS84 projection. Resolutions up to 25cm have been tested. 2m is recommended for a good trade off. Cellsize should be in degrees and co-ordinates must be in WGS84 decimal degrees.

To load multiple tiles, use commas eg. `-lid tile1.asc,tile2.asc`. You can load in different resolution tiles and use -resample to set the desired resolution (limited by data limits).

```
ncols        2454
nrows        1467
xllcorner   -1.475333294357
yllcorner   53.378635095801
cellsize     0.000006170864
NODATA_value 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
```

-udt User Defined CSV Point Clutter Files

This option is followed by a text file that allows you to define buildings with co-ordinates and heights. Elevations in the .udt file are evaluated and then copied into a temporary file under ***/tmp***. Then, the contents of the temp file are scanned, and if found to be unique, are added to the ground elevations described by the digital elevation data in memory. Height units are determined by appending M for meters or nothing for feet.

There is no special parsing for comments, so any lines of data that don't contain 3 numbers separated by commas are interpreted as a comment.

Format: latitude,longitude,height(M)

```
// This is a comment!
54.555,-2.221,10M
54.555,-2.222,10M
54.555,-2.223,10M
```

-ant Antenna Radiation Pattern Files

Antenna pattern data is read from a pair of files that default to having the same base name as the output file (`-o`), but with .az and .el extensions for azimuth and elevation patterns. This name can be overridden with the command line option `-ant` followed by a base filename or path and base filename for the .az and .el files. The program will first search for command line specified antenna files with the given `-ant` option, then it will default to searching for antenna files having the same name as the output file.

```
045.0
0       0.8950590
1       0.8966406
2       0.8981447
3       0.8995795
4       0.9009535
5       0.9022749
```

The first line of the .az file specifies the direction measured clockwise in degrees from True North. This is followed by azimuth headings (0 to 360 degrees) and their associated normalized field patterns (0.000 to 1.000) separated by whitespace. This direction can be overridden by the command line option `-rot` followed by a positive direction value from 0 to 360 degrees. This value will override any direction specified in the first line of the .az file. If not specified on the command line, the program will default to using the direction from the first line of the .az file.

The structure of SPLAT! elevation pattern files is slightly different. The first line of the .el file specifies the amount of mechanical beamtilt applied to the antenna. A downward tilt is expressed as a positive angle, while an upward tilt is expressed as a negative angle. This data is followed by the azimuthal direction of the tilt, 0 to 360 degrees, separated by whitespace. These can be overridden by the command line options `-dt` for downtilt and `-dtdir` for downtilt azimuthal direction. The options, like `-rot` above, will override any values embedded in the .el file if specified on the command line.

The remainder of the file consists of elevation angles and their radiation pattern (0.000 to 1.000) values separated by whitespace. Elevation angles must be specified over a -10.0 to +90.0 degree range. In this example, the antenna is tilted down 2 degrees towards an azimuth of 045.0 degrees.

```
2.0    045.0
-10.0   0.172
-9.5    0.109
-9.0    0.115
```

-8.5	0.155
-8.0	0.157

-clt MODIS 17-Class Wide Area Clutter File (In ASCII grid format):

This allows the user to specify a landcover clutter path and filename to use. The file is expected to be in the ASCII grid format (.asc), similar in format to the LIDAR tiles described above.

Supported classes of clutter:

0	WATER
1	EVERGREEN NEEDLELEAF FOREST
2	EVERGREEN BROADLEAF FOREST
3	DECIDUOUS NEEDLELEAF FOREST
4	DECIDUOUS BROADLEAF FOREST
5	MIXED FOREST
6	WOODLAND
7	WOODED GRASSLAND
8	CLOSED SHRUBLAND
9	OPEN SHRUBLAND
10	GRASSLAND
11	CROPLAND
12	BARE GROUND
13	URBAN AND BUILT-UP
15	No data

Running SignalServer – Command Line Options:

Usage: signalserver [data options] [input options] [antenna options] [output options] -o outputfile

```
Data:
     -sdf Directory containing SRTM derived .sdf DEM tiles (may be .gz or .bz2)
     -lid ASCII grid tile (LIDAR) with dimensions and resolution defined in header
     -udt User defined point clutter as decimal co-ordinates: 'latitude,longitude,height'
     -clt MODIS 17-class wide area clutter in ASCII grid format
     -color File to pre-load .scf/.lcf/.dcf for Signal/Loss/dBm color palette
Input:
     -lat Tx Latitude (decimal degrees) -70/+70
     -lon Tx Longitude (decimal degrees) -180/+180
     -rla (Optional) Rx Latitude for PPA (decimal degrees) -70/+70
     -rlo (Optional) Rx Longitude for PPA (decimal degrees) -180/+180
     -f Tx Frequency (MHz) 20MHz to 100GHz (LOS after 20GHz)
     -erp Tx Total Effective Radiated Power in Watts (dBd) inc Tx+Rx gain. 2.14dBi = 0dBd
     -gc Random ground clutter (feet/meters) Average 6m for suburban areas, or 2m for rural areas.
     -m Metric units of measurement
     -te Terrain code 1-6 (optional - 1. Water, 2. Marsh, 3. Farmland, 4. Mountain, 5. Desert, 6. Urban
     -terdic Terrain dielectric value 2-80 (optional)
     -tercon Terrain conductivity 0.01-0.0001 (optional)
     -cl Climate code 1-7 (optional – afffects signal attenuation - 1. Equatorial 2. Continental subtropical
3. Maritime subtropical 4. Desert 5. Continental temperate 6. Maritime temperate (Land) 7. Maritime
temperate (Sea)
     -rel Reliability for ITM model (% of 'time') 1 to 99 (optional, default 50%)
     -conf Confidence for ITM model (% of 'situations') 1 to 99 (optional, default 50%)
     -resample Reduce Lidar resolution by specified factor (2 = 50%)
Output:
     -o basename (Output file basename - required)
     -dbm Plot Rxd signal power instead of field strength in dBuV/m
     -rt Rx Threshold (dB / dBm / dBuV/m)
     -R Radius (miles/kilometers)
     -res Pixels per tile. 300/600/1200/3600 (Optional. LIDAR res is within the tile)
     -pm Propagation model (1-12) 1: ITM, 2: LOS, 3: Hata, 4: ECC33, 5: SUI, 6: COST-Hata, 7: FSPL,
8:ITWOM, 9: Ericsson, 10: Plane earth, 11: Egli VHF/UHF, 12: Soil
     -pe Propagation model mode (1-3) 1=Urban, 2=Suburban, 3=Rural
     -ked Knife edge diffraction (Already on for ITM)
Antenna:
     -ant (antenna pattern file path and basename for .az and .el files)
     -txh Tx Height (above ground)
     -rxh Rx Height(s) (optional. Default=0.1, Standard 1.83m or 6 feet)
     -rxg Rx gain dBd (optional for PPA text report, -2.14 =  0 dBi)
     -hp Horizontal Polarisation (default=vertical)
     -rot  (  0.0 - 359.0 degrees, default 0.0) Antenna Pattern Rotation
     -dt   ( -10.0 - 90.0 degrees, default 0.0) Antenna Downtilt
     -dtdir ( 0.0 - 359.0 degrees, default 0.0) Antenna Downtilt Direction
Debugging:
     -t Terrain greyscale background
```

-dbg Verbose debug messages
-ng Normalise Path Profile graph
-haf Halve 1 or 2 (optional)
-nothreads Turn off threaded processing

Running SignalServer - Scripting:

By using wrapper scripts like runsig.sh and genkmz.sh, you can streamline running SignalServer by pre-setting some commonly used options in the runsig.sh file. Those options should be the ones you use every time you run SignalServer, like "`-m`" for metric or "`-sdf ./sdf_file_path`", for example, so you won't have to specify those options every time you run it. The genkmz.sh file will convert the output ppm file into a .png with transparent background, then will convert it to a Google Earth Keyhole Markup Language (.kml) file, and then it compresses it for size to a (.kmz) file. For example, using the provided sample runsig.sh and genkmz.sh wrapper scripts generally goes like this:

```
sudo ./runsig.sh (-options...) -o outfile | ./genkmz.sh
```

This outputs outfile.ppm, outfile.png and outfile.kmz. It could be further automated to process this into calling from a database and then storing the resulting .kmz files. At about 2 minutes to produce a plot on a 4 core computer, one could model all 1400 repeaters in a given state in about 48 hours.

Running SignalServer - Examples:

90m resolution

- INPUTS: 900MHz tower at 25m AGL with 5W ERP, 30km radius
- OUTPUTS: 1200 resolution, 30km radius, -90dBm receiver threshold, Longley Rice model

```
 ./signalserver -sdf /data/SRTM3 -lat 51.849 -lon -2.2299 -txh 25 -f
900 -erp 5 -rxh 2 -rt -90 -dbm -m -o test1 -R 30 -res 1200 -pm 1
```

30m resolution

- INPUTS: 450MHz tower at 25f AGL with 20W ERP, 10km radius
- OUTPUTS: 3600 resolution, 30km radius, 10dBuV receiver threshold, Hata model

```
./signalserverHD -sdf /data/SRTM1 -lat 51.849 -lon -2.2299 -txh 25 -f
450 -erp 20 -rxh 2 -rt 10 -o test2 -R 10 -res 3600 -pm 3
```

2m resolution (LIDAR)

- INPUTS: 1800MHz tower at 15m AGL with 1W ERP, 1 km radius
- OUTPUTS: 2m LIDAR resolution, 5km radius, -90dBm receiver threshold, Longley Rice model

```
./signalserverLIDAR -lid /data/LIDAR/Gloucester_2m.asc -lat 51.849 -
lon -2.2299 -txh 15 -f 1800 -erp 1 -rxh 2 -rt -90 -dbm -m -o test3 -R
1 -pm 1
```

90m resolution interference contour

- INPUTS: 446MHz tower at 320f AGL with 700W ERP, 150 mile radius, 10% of time reliability.
- OUTPUTS: 1200 resolution, 21dBuV receiver threshold, Longley Rice model

```
./signalserver -sdf /mnt/data -lat 42.228889 -lon -87.812500 -txh 320
-rxh 6 -f 446.000 -erp 700 -rt 21 -te 3 -rel 10 -pm 1 -R 150 -color
color/green -o rpt_interf
```

30m resolution interference contour

- INPUTS: 446MHz tower at 320f AGL with 700W ERP, 150 mile radius, 10% of time reliability.
- OUTPUTS: 3600 resolution, 21dBuV receiver threshold, Longley Rice model

```
./signalserverHD -sdf /mnt/data-hd -lat 42.428889 -lon -87.812500 -
txh 320 -rxh 6 -f 446.000 -erp 700 -rt 21 -te 3 -rel 10 -pm 1 -R 150
-color -color/green -o rpt_intf-hd
```

30m resolution service contour

- INPUTS: 446MHz tower at 320f AGL with 700W ERP, 150 mile radius, 50% of time reliability.
- OUTPUTS: 1200 resolution, 39dBuV receiver threshold, Longley Rice model

```
./signalserver -sdf /mnt/data -lat 42.428889 -lon -87.812500 -txh 320
-rxh 6 -f 446.000 -erp 700 -rt 39 -te 3 -rel 50 -pm 1 -R 50 -color
color/blue -o rpt_srvc
```

90m resolution service contour
- INPUTS: 446MHz tower at 320f AGL with 700W ERP, 150 mile radius, 50% of time reliability.
- OUTPUTS: 3600 resolution, 39dBuV receiver threshold, Longley Rice model

```
./signalserverHD -sdf /mnt/data-hd -lat 42.428889 -lon -87.812500 -
txh 320 -rxh 6 -f 446.000 -erp 700 -rt 39 -te 3 -rel 50 -pm 1 -R 50 -
color color/blue -o rpt_srvc-hd
```

Running SignalServer with Wrapper Scripts - Examples:

30m resolution interference contour, DB413-B Antenna, azimuth at 225 deg:
- INPUTS: 446MHz tower at 320f AGL with 700W ERP, 150 mile radius, 10% of time reliability.
- OUTPUTS: 600 resolution, 21dBuV receiver threshold, Longley Rice model

```
sudo ./runsig.sh -lat 42.428889 -lon -87.812500 -txh 320 -f 446.000 -
erp 700 -R 150 -res 300 -rel 10 -rt 21 -ant antenna/DB413-B -rot 225
-color color/green -o rpt_interf | ./genkmz.sh
```

30m resolution service contour. DB413-B Antenna, azimuth at 225 deg:
- INPUTS: 446MHz tower at 320f AGL with 700W ERP, 150 mile radius, 50% of time reliability.
- OUTPUTS: 600 resolution, 39dBuV receiver threshold, Longley Rice model

```
sudo ./runsig.sh -lat 42.428889 -lon -87.812500 -txh 320 -f 446.000 -
erp 700 -R 150 -res 300 -rel 50 -rt 39 -ant antenna/DB413-B -rot 225
-color color/blue -o rpt_srvc | ./genkmz.sh
```

UDT Clutter File Example:
- INPUTS: 446MHz tower at 320f AGL with 700W ERP, 150 mile radius, 50% of time reliability.
- OUTPUTS: 600 resolution, 39dBuV receiver threshold, Longley Rice model

```
sudo ./runsig.sh -lat 42.428889 -lon -87.812500 -txh 320 -f 446.000 -
erp 700 -R 150 -res 600 -rel 50 -rt 39 -ant antenna/DB413-B -rot 225
-color color/blue -o rpt_clutter | ./genkmz.sh
```

Running SignalServer - Misc Options/Notes:

Variability Modes of the Longley-Rice Model:

The way most RF coverage prediction programs work (SignalServer included) is to draw multiple point-to-point radials around a transmitter to the "receiver" unit, rotating around the given transmit location. Each radial is traced out to it's end point; it determines where that is by what the average signal strength is along the ray it's casting. The end point will be mostly determined by when the signal strength drops below the threshold value given with (`-rt`). Once it reaches that approximate endpoint, it mathematically simulates jittering about and taking a bunch of random samples. Then, some statistical average weighting is applied to those, which are the "variability modes".

To more accurately simulate the real world effects of propagation physics, the calculations ultimately come up with a statistical value of probability that in a given set of of situations, locations and times, that a given signal will be above or below a set threshold value (-rt).

The Specific Variability Mode Percentages:

Times = samples multiple times per day, per hour, per minute, etc. This value defaults to 50%. This is best thought of as "50% of the times, the signal will be at or greater than the threshold value". This most directly affects availability or transmission reliability. Fade margins are expressed this way, and typical design models call for 18 dB of fade margin, so that fits in at about a 98% reliability factor:

50%	6 dB
90%	10 dB
99%	20 dB
99.9%	30 dB
99.99%	40 dB

Time is the primary variability that affects the outcome of the ultimate go-no-go decision of whether a signal is above or below the threshold value, so it has the most effect on the outcome of the plots. It is usually the only value that is changed when plotting either service and interference contours, the others usually are at their default values of 50%.

Locations = samples multiple locations around the end point. Uses many jittered positions, so it takes an average around the area, and it is expressed as a percentage of sample locations at which the signal will be at or above a given value. Note that is value is copied from either the Times or Situations percentage values given, and is not directly accessible from the command line. It is typically specified at it’s default value of 50%

Situations = random stuff - Acts of God, Brownian Motion, Cosmic Rays, truly random events that can affect things. These natural effects are always present, so they affect the ultimate outcome of the variability modes. This is usually assumed to be at its default baseline of 50% and is dropped from the common notation of the F(X, Y) variability mode notation because it's relatively unimportant and does not change (much).

The variability is usually expressed as F(LOCATIONS,TIME). When doing TSB-88 style interference and service contours, the typical values used are F(50, 50) for plotting service contours and F(50, 10) for plotting interference contours. SignalServer uses the naming convention of signal “reliablity” and “confidence”. Reliability directly refers to “Times” and “Confidence” refers directly to “Situations”:

-rel reliability in LR model [LR.rel] (% of TIME F(50, rel) param, default=50)
-conf confidence in LR model [LR.conf] (% of SITUATIONS F(conf, 50) param, default=50)

For the usual use, F(50, 10) for service plots uses the default of 50% for situations, and that value is copied to locations as well. Times (`-rel`) is set to 10%. For interference plots nothing needs to be specified at all to assume the default values of 50% for all three.