CRC-COVLIB API Reference

Communications Research
Centre Canada

November 21, 2025

Contents

CRC-COVLIB is a basic application programming interface (API) for radio coverage prediction developed at the Communications Research Centre Canada. It includes a few different propagation models:

- The Longley-Rice propagation model developed at the US National Telecommunications and Information Administration (NTIA).
- The Extended-Hata (eHata) urban propagation model also from the NTIA.
- An implementation of the ITU-R P.1812 Recommendation from the International Telecommunication Union (ITU).
- An implementation of the ITU-R P.452 Recommendation also from the ITU.
- CRC-MLPL, a machine learning-based path loss model developed at the Communications Research Centre Canada, trained with an openly available ITU-R UK Ofcom drive test dataset.[1]

This document details the C++ programming interface that is made available by CRC-COVLIB. It does not go into the details of radio wave propagation science and transmission/reception technologies.

A Python wrapper is also available in order to easily invoke CRC-COVLIB from the Python programming language. No alternate API documentation is currently available for Python, however this document may still serve as a good reference as method signatures have been preserved for the Python interface.

Code snippet

The CRC-COVLIB library exports a function named **`NewSimulation`** that returns a pointer to an object implementing the **`ISimulation`** interface. Typically, this object would first be used to set simulation parameters (i.e. transmitter parameters, receiver parameters, terrain elevation data sources, etc.) before calling the coverage prediction algorithm. Once done using the object, its **`Release`** method may be called to destroy it and free resources. Below is a small code example using CRC-COVLIB's functionalities:

```
#include "CRC-COVLIB.h"
using namespace Crc::Covlib;

int main(int argc, char* argv[])
{
  ISimulation* sim = NewSimulation();
  if( sim != NULL )
  {
    // sets some transmitter parameters...
    sim->SetTransmitterLocation(45.1, -75.2);

    // selects a propagation model
    sim->SetPropagationModel(LONGLEY_RICE);

    // sets parameters related to the Longley-Rice propagation model...
    sim->SetLongleyRiceTimePercentage(50.0);

    // sets some receiver parameters...
    sim->SetReceiverHeightAboveGround(3.0);

    // specifies the directory containing the terrain elevation data files
    sim->SetPrimaryTerrainElevDataSource(TERR_ELEV_NRCAN_CDEM);
    sim->SetTerrainElevDataSourceDirectory(TERR_ELEV_NRCAN_CDEM, "C:/Data/CDEM");
    sim->SetTerrainElevDataSamplingResolution(50);

    // specifies the geographic area where the signal should be estimated
    sim->SetReceptionAreaCorners(44.5, -76.0, 45.5, -74.0);

    // selects a result type
    sim->SetResultType(FIELD_STRENGTH_DBUVM);

    // runs the simulation (coverage prediction algorithm)
    sim->GenerateReceptionAreaResults();

    // writes results into a text file and a raster file
    sim->ExportReceptionAreaResultsToTextFile("C:/temp/simResults.txt");
    sim->ExportReceptionAreaResultsToBilFile("C:/temp/simResults.bil");

    sim->Release(); // destroys the sim instance, free resources
  }

  return 0;
}
```

The Crc::Covlib namespace

In addition to various `enum` types that are described later in this document, the `Crc::Covlib` namespace contains 3 functions.

- `ISimulation* NewSimulation()`

Returns a pointer to a newly created object implementing the `ISimulation` interface.

- `ISimulation* DeepCopySimulation(ISimulation* sim)`

Also creates a new object implementing the `ISimulation` interface. The new object is a deep copy of the object specified as parameter.

- `bool SetITUProprietaryDataDirectory(const char* directory)`

Simulation objects created by **`NewSimulation`** and **`DeepCopySimulation`** may need to use digital map files from the International Telecommunication Union (ITU) if they are using any of the ITU-R P.1812, ITU-R P.452 and ITU-R P.676 models. **`SetITUProprietaryDataDirectory`** should be employed to indicate where to find these ITU digital maps. More precisely, CRC-COVLIB is looking for these four files:

1. DN50.TXT from https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.1812-7-202308-I!!ZIP-E.zip
2. N050.TXT from https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.1812-7-202308-I!!ZIP-E.zip
3. T_Annual.TXT from https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.1510-1-201706-I!!ZIP-E.zip
4. surfwv_50_fixed.txt from https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.2001-5-202308-I!!ZIP-E.zip

Since our request to redistribute those files or to incorporate them within CRC-COVLIB's source code was kindly declined by the ITU, they need to be downloaded from their original website. Permission to create a script to download them was however obtained given 3 conditions: that the files' URLs are displayed as they are downloaded, that a notice is included mentioning that those files are for personal use only, and that the files are left in their original text (.txt) format. Hence they may either be installed manually or by using the provided **install_ITU_data.py** python script.

If the ITU files are not made available to CRC-COVLIB, simulations using ITU models can still be run. In that case, a reasonable default value will be used in place of the location-specific values that would have otherwise been obtained from the digital map files (see **`GetITUDigitalMapValue`** for details on the default values). Another option is to explicitly specify the values to use for the simulation (for example using **`SetITURP1812AverageRadioRefractivityLapseRate`** or **`SetITURP1812SeaLevelSurface-Refractivity`** when using the ITU-R P.1812 model).

`SetITUProprietaryDataDirectory` will return `true` when all four files are found and read successfully, `false` otherwise.

C++ API vs Python wrapper API

Before continuing with the different methods of the `ISimulation` interface in the sections ahead, it is worth mentioning a few words on the very few particularities of the Python wrapper API compared to the C++ API.

In Python the **`NewSimulation`** function is called from the constructor (i.e. the **`__init__`**() method) of the **`Simulation`** class from the **`simulation.py`** module. A new instance of the **`Simulation`** class should therefore be created instead of calling **`NewSimulation`**. The **`Release`** method also does not need to be called as it is invoked from the destructor (i.e. the **`__del__`**() method) of the class.

Copies of **`Simulation`** objects can be made through the use of the standard **`copy`** Python module.

Finally, **`SetITUProprietaryDataDirectory`** is automatically called when importing the **`simulation.py`** module. It is called with the "crc_covlib/data/itu_proprietary/" directory as input, where it expects to find the ITU digital map files. If they are not found or could not be successfully read, a warning message will be displayed in the console. Here is a small Python code example:

```python
from crc_covlib import simulation # calls SetITUProprietaryDataDirectory
import copy

if __name__ == '__main__':
  sim = simulation.Simulation() # creates new Simulation object (calls NewSimulation)
  sim.SetTransmitterLocation(45.1, -75.2)
  ...
  sim_copy = copy.deepcopy(sim) # creates a deep copy of sim
  ...
  exit() # sim object is destroyed here (Release is called on the sim object)
```

Transmitter methods

- `void SetTransmitterLocation(double latitude_degrees, double longitude_degrees)`
- `double GetTransmitterLatitude()`
- `double GetTransmitterLongitude()`

Sets/gets transmitter location in degrees of latitude and degrees of longitude. Northern hemisphere latitudes must be positive while Southern hemisphere latitudes must be negative. Similarly, East longitudes must be positive while West longitudes must be negative. Valid range of latitudes is from -90 to 90 degrees, valid range of longitudes is from -180 to 180 degrees. Default transmitter location is 45 degrees North (+45), 75 degrees West (-75).

Note that here and everywhere in this document, when latitude and longitude parameters are mentioned, "WGS 84" should be assumed as the coordinate reference system.

- `void SetTransmitterHeight(double height_meters)`
- `double GetTransmitterHeight()`

Sets/gets the height of the transmitter's radiation center above ground level, in meters. Valid range of heights for the Longley-Rice propagation model is from 0.5 to 3000 meters. Valid range of heights for the ITU-R P.1812 propagation model is from 1 to 3000 meters. ITU-R P.452 does not define any range limits for the transmitter height, and valid range of heights for the eHata propagation model is from 30 to 200 meters. **`SetTransmitterHeight`** only accepts values between 0.5 and 3000 meters. Default value is 50 meters.

- `void SetTransmitterFrequency(double frequency_MHz)`
- `double GetTransmitterFrequency()`

Sets/gets the transmitter frequency, in megahertz. *`frequency_MHz`* may be any value greater than zero, however different propagation models have different ranges of suitable frequencies. Valid range of frequencies is from 20 to 20000 MHz for the Longley-Rice propagation model, 30 to 6000 MHz for the ITU-R P.1812 propagation model and 100 to 50000 MHz for the ITUR-P.452 propagation model. The eHata propagation model should be suitable for frequencies from about 100 MHz to 3500 MHz. Default value for the transmitter frequency is 100 MHz.

- `void SetTransmitterPower(double power_watts, PowerType powerType=EIRP)`
- `double GetTransmitterPower(PowerType powerType=EIRP)`

Sets/gets the transmitter power, in watts. `PowerType` is an enumeration type that may take one of the following values:

```
TPO  = 1
ERP  = 2
EIRP = 3
```

The power may be specified or obtained as a transmit power output value (**TPO**), an effective radiated power value (**ERP**) or an effective isotropic radiated power value (**EIRP**). The specified power value must be greater than zero.

The following relationships exist between the power types:

- `EIRP (dBW) = TPO (dBW) – L (dB) + G (dBi)`
- `EIRP (dBW) = ERP (dBW) + 2.15`

```
where
 L is the transmitter losses, in dB (see SetTransmitterLosses method)
 G is the transmitter antenna maximum gain, in dBi (see SetAntennaMaximumGain method)
```

The transmitter power is only considered for the **FIELD_STRENGTH_DBUVM** and **RECEIVED_POWER_DBM** result types (see **SetResultType** method).

Default value is 1000 watts of EIRP.

- `void SetTransmitterLosses(double losses_dB)`
- `double GetTransmitterLosses()`

Sets/gets the losses, in dB, associated with the transmitter (cable, connectors, etc.). The losses should be entered as a positive value. Transmitter losses are not considered for the **PATH_LOSS_DB** result type (see **SetResultType** method). Default value is 0 dB.

- `void SetTransmitterPolarization(Polarization polarization)`
- `Polarization GetTransmitterPolarization()`

Sets/gets transmitter polarization. `Polarization` is an enumeration type that may take one of the following values:

```
HORIZONTAL_POL = 0
VERTICAL_POL   = 1
```

Default value is **VERTICAL_POL**.

Receiver methods

- `void SetReceiverHeightAboveGround(double height_meters)`
- `double GetReceiverHeightAboveGround()`

Sets/gets the receiver height above ground level, in meters. Valid range of heights for the Longley-Rice propagation model is from 0.5 to 3000 meters. Valid range of heights for the ITU-R P.1812 propagation model is from 1 to 3000 meters. ITU-R P.452 does not define any range limits for the receiver height, and valid range of receiver heights for the eHata propagation model is from 1 to 10 meters. **`SetReceiverHeightAboveGround`** only accepts values between 0.5 and 3000 meters. Default value is 1.5 meters.

- `void SetReceiverLosses(double losses_dB)`
- `double GetReceiverLosses()`

Sets/gets the losses, in dB, associated with the receiver (cable, connectors, etc.). The losses should be entered as a positive value. Receiver losses are only considered for the **`TRANSMISSION_LOSS_DB`** and **`RECEIVED_POWER_DBM`** result types (see **`SetResultType`** method). Default value is 0 dB.

Antenna methods

Antenna parameters may or may not be used in a simulation depending on other input parameters.

- Antenna parameters at the transmitter are considered for the **`FIELD_STRENGTH_DBUVM`**, **`TRANSMISSION_LOSS_DB`** and **`RECEIVED_POWER_DBM`** result types (see **`SetResultType`** method).
- Antenna parameters at the receiver are considered for the **`TRANSMISSION_LOSS_DB`** and **`RECEIVED_POWER_DBM`** result types.
- Additionally, antenna parameters at both the transmitter and receiver are considered by the ITU-R P.452 propagation model for the calculation of tropospheric scattering losses.

Every antenna related method below takes a `Terminal` type parameter as input. `Terminal` is an enumeration type that may take one of the following values:

```
TRANSMITTER = 1
RECEIVER    = 2
```

- `void AddAntennaHorizontalPatternEntry(Terminal terminal, double azimuth_degrees, double gain_dB)`

Adds a new gain (in dB) entry at the specified azimuth (in degrees) in the horizontal pattern of an antenna.

The *`terminal`* parameter determines whether the method applies to the **`TRANSMITTER`** or **`RECEIVER`** antenna. The *`azimuth_degrees`* parameter must be between 0 and 359.99 degrees, where 0 degree should be used for the direction of maximum gain (main lobe). The gain values should be normalized (i.e. 0 dB in the direction of maximum gain, other gain values are negative). If non-normalized gain values are added to the pattern, they can be normalized afterwards once all entries have been added using the **`NormalizeAntennaHorizontalPattern`** method.

By default the horizontal antenna pattern contains no entry for both terminals. In that situation, the horizontal antenna pattern gain is assumed to be 0 dB at every azimuth.

- `void AddAntennaVerticalPatternEntry(Terminal terminal, int azimuth_degrees, double elevAngle_degrees, double gain_dB)`

Adds a new gain (in dB) entry at the specified azimuth (in degrees) and elevation angle (in degrees) in the vertical pattern of an antenna.

The *`terminal`* parameter determines whether the method applies to the **`TRANSMITTER`** or **`RECEIVER`** antenna. The *`azimuth_degrees`* parameter allows to enter different vertical pattern “slices”. It must be an integer between 0 and 359 degrees inclusively, where 0 degree should be used for the direction of maximum gain (main lobe). The *`elevAngle_degrees`* parameter must be between -90 and +90 degrees inclusively, where -90 is the zenith, 0 is the astronomical horizon and +90 is the nadir. The gain values should be normalized (i.e. 0 dB in the direction of maximum gain, other gain values are negative). If non-normalized gain values are added to the pattern, they can be normalized afterwards once all entries have been added using the **`NormalizeAntennaVerticalPattern`** method.

In real life situations, it is common to only have a 2D antenna pattern (one horizontal plane plus one vertical plane) available. In that case the vertical antenna pattern data should be entered at 0 degree (front lobe vertical slice) and possibly at 180 degrees (back lobe vertical slice) of azimuth. Depending on the approximation method used (see **`SetAntennaPatternApproximationMethod`**), the back lobe vertical slice may be ignored in simulations.

When a full 3D pattern is available, the best solution may be to enter the data as multiple vertical slices without using a horizontal pattern (that is, using **`AddAntennaVerticalPatternEntry`** with different azimuths as input but without using **`AddAntennaHorizontalPatternEntry`**). The approximation method should then be set to **`V_PATTERN_ONLY`** for best results and to ensure all slices will be considered.

By default the vertical antenna pattern contains no entry for both terminals. In that situation, the vertical antenna pattern gain is assumed to be 0 dB at every angle.

- `void` **`ClearAntennaPatterns`**`(Terminal` *`terminal`*`, bool` *`clearHorizontalPattern`*`=true, bool` *`clearVerticalPattern`*`=true)`

Deletes all entries from the horizontal pattern (if *`clearHorizontalPattern`* is `true`) and vertical pattern (if *`clearVerticalPattern`* is `true`) of an antenna. The *`terminal`* parameter determines whether the method applies to the **`TRANSMITTER`** or **`RECEIVER`** antenna.

- `void` **`SetAntennaElectricalTilt`**`(Terminal` *`terminal`*`, double` *`elecricalTilt_degrees`*`)`
- `double` **`GetAntennaElectricalTilt`**`(Terminal` *`terminal`*`)`

Sets/gets the electrical tilt, in degrees, of an antenna.

The *`terminal`* parameter determines whether the method applies to the **`TRANSMITTER`** or **`RECEIVER`** antenna. The *`elecricalTilt_degrees`* parameter must be between -90 and +90 degrees inclusively, where -90 is the zenith, 0 is the astronomical horizon and +90 is the nadir. The tilt value should be set to 0 when the desired tilt is already included in the antenna pattern.

Default value is 0 degree of electrical tilt for both terminals.

- `void` **`SetAntennaMechanicalTilt`**`(Terminal` *`terminal`*`, double` *`mechanicalTilt_degrees`*`, double` *`azimuth_degrees`*`=0)`
- `double` **`GetAntennaMechanicalTilt`**`(Terminal` *`terminal`*`)`
- `double` **`GetAntennaMechanicalTiltAzimuth`**`(Terminal` *`terminal`*`)`

Sets/gets the mechanical tilt, in degrees, of an antenna.

The *terminal* parameter determines whether the method applies to the **TRANSMITTER** or **RECEIVER** antenna. The *azimuth_degrees* parameter is the azimuth of the horizontal pattern at which the mechanical tilt will be applied. It must be between 0 and 359.99 degrees, where 0 degree would normally be used for the direction of maximum gain (main lobe). Therefore the *azimuth_degrees* parameter should ordinarily be set to 0, although it is possible to use a different value if desired. The *mechanicalTilt_degrees* parameter must be between -90 and +90 degrees inclusively, where -90 is the zenith, 0 is the astronomical horizon and +90 is the nadir. The tilt value should be set to 0 when the desired tilt is already included in the antenna pattern.

Default value is 0 degree of mechanical tilt for both terminals.

- `void SetAntennaMaximumGain(Terminal terminal, double maxGain_dBi)`
- `double GetAntennaMaximumGain(Terminal terminal)`

Sets/gets the maximum gain of an antenna, in decibels referenced to the gain of a theoretical isotropic radiator.

The *terminal* parameter determines whether the method applies to the **TRANSMITTER** or **RECEIVER** antenna. Default value is 0 dBi for both terminals.

- `void SetAntennaBearing(Terminal terminal, BearingReference bearingRef, double bearing_degrees)`
- `BearingReference GetAntennaBearingReference(Terminal terminal)`
- `double GetAntennaBearing(Terminal terminal)`

Set/gets the bearing of an antenna, in degrees from the specified reference.

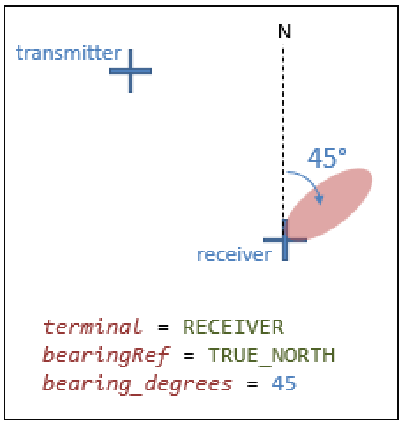
The *terminal* parameter determines whether the method applies to the **TRANSMITTER** or **RECEIVER** antenna. **BearingReference** is an enumeration type that may take one of the following values:

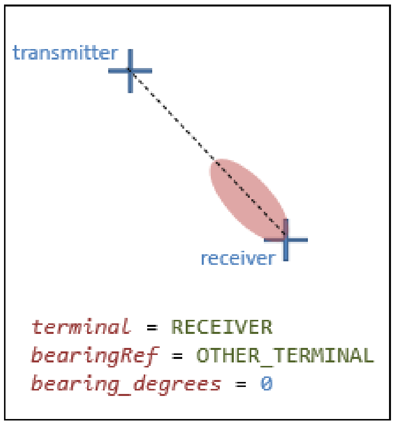
```
TRUE_NORTH     = 1
OTHER_TERMINAL = 2
```

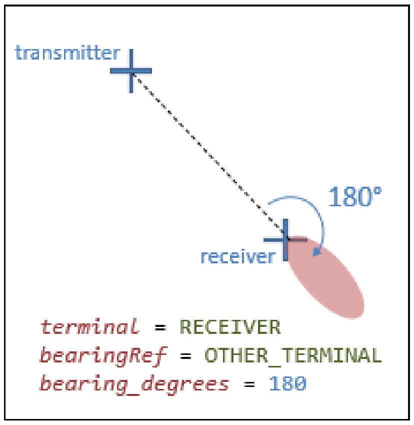
The *bearing_degrees* parameter must be between 0 and 359.99 degrees. When the bearing reference is set to **TRUE_NORTH** and the bearing to 0 degree, the antenna's main lobe will point towards true north (assuming the main lobe was entered at 0 degree of azimuth when using the **AddAntennaHorizontal-PatternEntry** and/or **AddAntennaVerticalPatternEntry** methods). A bearing of 90 degrees will have the antenna point towards east, 180 degrees towards south and 270 degrees towards west.

When the bearing reference is set to **OTHER_TERMNAL** and the bearing to 0 degree, the antenna's main lobe will point towards the other terminal (i.e. towards the receiver for the transmitter antenna and towards the transmitter for the receiver antenna). Increasing the bearing rotates the main lobe clockwise from that direction (e.g. 180 degrees of bearing will have the antenna pointing away from the other terminal).

Note that the bearing parameters do not affect the vertical angle at which an antenna is pointing.







Default value is 0 degree from **TRUE_NORTH** for the **TRANSMITTER** antenna.
Default value is 0 degree from the **OTHER_TERMINAL** for the **RECEIVER** antenna.

- double **NormalizeAntennaHorizontalPattern**(Terminal *terminal*)
- double **NormalizeAntennaVerticalPattern**(Terminal *terminal*)

Normalizes the horizontal or vertical pattern of an antenna.

The *terminal* parameter determines whether the method applies to the **TRANSMITTER** or **RECEIVER** antenna. After pattern entries have been added using the **AddAntennaHorizontalPatternEntry** and/or **AddAntennaVerticalPatternEntry** methods, patterns can be normalized using the **NormalizeAntennaHorizontalPattern** and/or **NormalizeAntennaVerticalPattern** methods. Normalizing a pattern brings its highest gain entry to 0 dB and adjusts all other entries by the same amount the highest gain was modified. The return value is this amount, in dB, by which the gain entries were modified. It can either be a positive amount (when the maximum gain entry was lower than 0 dB) or a negative one (when the maximum gain entry was higher than 0 dB). Note that the maximum antenna gain (in dBi) accessible from **Get/SetAntennaMaximumGain** is not modified when a pattern is normalized. The **SetAntennaMaximumGain** must be called explicitly in order to adjust the maximum gain (in dBi), if required, after the normalization.

- void **SetAntennaPatternApproximationMethod**(Terminal *terminal*,
  PatternApproximationMethod *method*)
- PatternApproximationMethod **GetAntennaPatternApproximationMethod**(
  Terminal *terminal*)

Sets/gets the 3D approximation method (sometimes called the *interpolation method*) for obtaining the antenna gain in any direction from the horizontal and vertical patterns.

The *terminal* parameter determines whether the method applies to the **TRANSMITTER** or **RECEIVER** antenna. PatternApproximationMethod is an enumeration type that may take one of the following values:

H_PATTERN_ONLY	= 1
V_PATTERN_ONLY	= 2
SUMMING	= 3

```
WEIGHTED_SUMMING = 4
HYBRID           = 5
```

The **H_PATTERN_ONLY** method uses the horizontal pattern only while the **V_PATTERN_ONLY** uses the vertical pattern only (i.e. all available vertical slices).

The **SUMMING** method is a common approximation method that simply sums the horizontal pattern gain (in dB) at the desired azimuth with the vertical pattern gain (in dB) at the desired vertical angle. Other methods like **WEIGHTED_SUMMING** (see [3]) and **HYBRID** (see [4]) are a little more complex and are usually more accurate. These methods make use of the horizontal pattern and the front lobe vertical slice only (i.e. the vertical pattern data entered at 0 degree of azimuth only). Any other vertical slice (including the back lobe vertical slice) are ignored. These methods also generally assume that pattern entries have been normalized (see **NormalizeAntennaHorizontalPattern** and **NormalizeAntennaVertical-Pattern**).

When generating results for a simulation, antenna gains may be needed or not depending on the selected result type (see **SetResultType**) and propagation model. If required, the horizontal pattern gain is obtained using the azimuth of the great-circle path between the transmitter and receiver locations. For the vertical pattern gain, the required vertical angle is computed using an algorithm from the *Path profile analysis* section of recommendations ITU-R P.1812 and ITU-R P.452 (even when a non-ITU propagation model is selected for the simulation). This algorithm uses the antenna heights above ground level, the terrain elevation profile between the terminals (if one or more terrain elevation data source(s) were specified for the simulation) and the average radio-refractivity lapse-rate of the atmosphere. In line-of-sight situations, the computed vertical angle will be towards the other terminal's antenna, otherwise it will be towards the radio horizon (i.e. the terrain clearance angle).

Default value for the approximation method is **HYBRID** for both the **TRANSMITTER** and **RECEIVER** terminals.

- `double GetAntennaGain(Terminal terminal, double azimuth_degrees,`
  `double elevAngle_degrees, double receiverLatitude_degrees=0,`
  `double receiverLongitude_degrees=0)`

Gets the approximated gain, in dBi, of an antenna at the specified azimuth and elevation angle.

The *terminal* parameter determines whether the method applies to the **TRANSMITTER** or **RECEIVER** antenna. **GetAntennaGain** uses the approximation method specified using **SetAntennaPattern-ApproximationMethod**. It also uses patterns, maximum gain, tilts and bearing parameters.

An *azimuth_degrees* value of 0 (or any multiple of 360) is interpreted as true north. The *elevAngle_degrees* parameter must be between -90 and +90 degrees inclusively, where -90 is the zenith, 0 is the astronomical horizon and +90 is the nadir. When the terminal's bearing reference (see **SetAntennaBearing**) is set to **OTHER_TERMINAL**, the receiver's coordinates are required in order to correctly evaluate the impact of the antenna's bearing, otherwise *receiverLatitude_degrees* and *receiverLongitude_degrees* are not used.

Propagation model selection methods

- `void SetPropagationModel(PropagationModel propagationModel)`
- `PropagationModel GetPropagationModel()`

Sets/gets the propagation model to use when generating results. `PropagationModel` is an enumeration type that may take one of the following values:

```
LONGLEY_RICE      = 0
ITU_R_P_1812      = 1
ITU_R_P_452_V17   = 2
ITU_R_P_452_V18   = 3
FREE_SPACE        = 4
EXTENDED_HATA     = 5
CRC_MLPL          = 6
CRC_PATH_OBSCURA  = 7
```

`LONGLEY_RICE` corresponds to the Irregular Terrain Model (ITM), also known as Longley-Rice (from NTIA).

`ITU_R_P_1812` corresponds to recommendation ITU-R P.1812-7.

`ITU_R_P_452_V17` corresponds to recommendation ITU-R P.452-17.

`ITU_R_P_452_V18` corresponds to recommendation ITU-R P.452–18.

`FREE_SPACE` corresponds to free space transmission (see recommendation ITU-R P.525).

`EXTENDED_HATA` corresponds to the Extended-Hata (eHata) urban propagation model (from NTIA).

`CRC_MLPL` is a machine learning-based path loss model developed at the Communications Research Centre Canada (2024), trained with a publicly available ITU-R UK Ofcom drive test dataset.[1]

`CRC_PATH_OBSCURA` is the latest machine learning-based path loss model developed at the Communications Research Centre Canada (2025), trained with a publicly available ITU-R UK Ofcom drive test dataset.[2]

Default value for the propagation model is `LONGLEY_RICE`.

	Frequency (MHz)	Tx height (m)	Rx height (m)	Path length (km)	Terrain elevation data usage	Surface elevation data usage	Land cover data usage
Longley-Rice	20-20,000	0.5-3,000	0.5-3,000	1-2,000	Yes	No	No
ITU-R P.1812	30-6,000	1-3,000	1-3,000	0.25-3,000	Yes	Yes[1]	Yes[1]
ITU-R P.452	100-50,000	N/A[2]	N/A[2]	N/A[2]	Yes	Yes[3]	Yes[3]
Free space	N/A	N/A	N/A	N/A	Yes[4]	No	No
eHata	100-3,500	30-200	1-10	1-100	Yes	No	No
CRC-MLPL	500-6,000[5]	N/A	N/A	0.050-50	Yes[4]	Yes	No
CRC Path Obscura	500-6,000[5]	N/A	N/A	0.050-50	Yes	Yes	No

1 Uses either surface or land cover data (see SetITURP1812SurfaceProfileMethod).

2 It should be reasonable to assume similar ranges to those from ITU-R P.1812.

3 Version 17 of ITU-R P.452 does not use surface data, it may only use land cover data (see SetITURP452Height-GainModelMode). Version 18 uses either surface or land cover data (see SetITURP452SurfaceProfileMethod).

4 Only used at transmitter and receiver locations.

5 500 to 6,000 MHz for best performance, but expected to yield acceptable results at up to 10,000 MHz.

Longley-Rice propagation model methods

- `void SetLongleyRiceSurfaceRefractivity(double refractivity_NUnits)`
- `double GetLongleyRiceSurfaceRefractivity()`

Sets/gets the surface refractivity, in N-units. Valid range is from 250 to 400 N-units. Default value is 301 N-units.

- `void SetLongleyRiceGroundDielectricConst(double dielectricConst)`
- `double GetLongleyRiceGroundDielectricConst()`

Sets/gets dielectric constant of ground. Valid range is from 4 to 81. Default value is 15.

- `void SetLongleyRiceGroundConductivity(double groundConduct_Sm)`
- `double GetLongleyRiceGroundConductivity()`

Sets/gets the conductivity of ground, in Siemens per meter. Valid range is from 0.001 to 5 S/m. Default value is 0.005 S/m.

- `void SetLongleyRiceClimaticZone(LRClimaticZone climaticZone)`
- `LRClimaticZone GetLongleyRiceClimaticZone()`

Sets/gets the climatic zone. `LRClimaticZone` is an enumeration type that may take one of the following values:

```
LR_EQUATORIAL                   = 1
LR_CONTINENTAL_SUBTROPICAL      = 2
LR_MARITIME_SUBTROPICAL         = 3
LR_DESERT                       = 4
LR_CONTINENTAL_TEMPERATE        = 5
LR_MARITIME_TEMPERATE_OVER_LAND = 6
LR_MARITIME_TEMPERATE_OVER_SEA  = 7
```

Default value is `LR_CONTINENTAL_TEMPERATE`.

- `void SetLongleyRiceActivePercentageSet(LRPercentageSet percentageSet)`
- `LRPercentageSet GetLongleyRiceActivePercentageSet()`

Sets/gets the set of percentage values to be used for path loss calculation. Longley-Rice may either use the time/location/situation set or the confidence/reliability set. `LRPercentageSet` is an enumeration type that may take one of the following values:

```
LR_TIME_LOCATION_SITUATION = 1
LR_CONFIDENCE_RELIABILITY  = 2
```

Default value is `LR_TIME_LOCATION_SITUATION`.

- `void SetLongleyRiceTimePercentage(double time_percent)`
- `double GetLongleyRiceTimePercentage()`

Sets/gets the time percentage (i.e. time variability). Valid range is from 0.1 to 99.9. Default value is 50. The time percentage may only apply when the active percentage set is `LR_TIME_LOCATION_SITUATION`.

- `void SetLongleyRiceLocationPercentage(double location_percent)`
- `double GetLongleyRiceLocationPercentage()`

Sets/gets the location percentage (i.e. location variability). Valid range is from 0.1 to 99.9. Default value is 50. The location percentage may only apply when the active percentage set is `LR_TIME_LOCATION_SITUATION`.

- `void SetLongleyRiceSituationPercentage(double situation_percent)`
- `double GetLongleyRiceSituationPercentage()`

Sets/gets the situation percentage (i.e. situation variability). Valid range is from 0.1 to 99.9. Default value is 50. The situation percentage only applies when the active percentage set is `LR_TIME_LOCATION_SITUATION`.

- `void SetLongleyRiceConfidencePercentage(double confidence_percent)`
- `double GetLongleyRiceConfidencePercentage()`

Sets/gets the confidence percentage (i.e. confidence variability). Valid range is from 0.1 to 99.9. Default value is 50. The confidence percentage may only apply when the active percentage set is `LR_CONFIDENCE_RELIABILITY`.

- `void SetLongleyRiceReliabilityPercentage(double reliability_percent)`
- `double GetLongleyRiceReliabilityPercentage()`

Sets/gets the reliability percentage (i.e. reliability variability). Valid range is from 0.1 to 99.9. Default value is 50. The reliability percentage may only apply when the active percentage set is `LR_CONFIDENCE_RELIABILITY`.

- `void SetLongleyRiceModeOfVariability(int mode)`
- `int GetLongleyRiceModeOfVariability()`

Sets/gets the mode of variability. The following values may be used for *mode*:

  0 = single message mode

1 = accidental mode
2 = mobile mode
3 = broadcast mode

Additionally, 10 may be added to *mode* for the location variability to be eliminated, and 20 may be added for the situation variability to be eliminated.

Consequently, valid values for *mode* are from 0 to 3 inclusively, 10 to 13 inclusively, 20 to 23 inclusively and 30 to 33 inclusively. Default value is 12.

Alternately, values from **LRModeOfVariability** may be used to specify the mode of variability instead of directly using an integer value. **LRModeOfVariability** is an enumeration type containing the following values:

```
LR_SINGLE_MESSAGE_MODE              = 0
LR_ACCIDENTAL_MESSAGE_MODE          = 1
LR_MOBILE_MODE                      = 2
LR_BROADCAST_MODE                   = 3
LR_ELIMINATE_LOCATION_VARIABILITY   = 10
LR_ELIMINATE_SITUATION_VARIABILITY  = 20
```

For example:

```
SetLongleyRiceModeOfVariability(LR_MOBILE_MODE + LR_ELIMINATE_LOCATION_VARIABILITY)
```

Additional resources

Additional resources regarding the Longley-Rice / ITM (Irregular Terrain Model) propagation model are available at:

https://github.com/NTIA/itm
https://its.ntia.gov/research-topics/radio-propagation-software/itm/itm/
https://udel.edu/~mm/itm/

ITU-R P.1812 propagation model methods

- `void SetITURP1812TimePercentage(double time_percent)`
- `double GetITURP1812TimePercentage()`

Sets/gets the percentage of average year for which the calculated basic transmission loss is not exceeded. Valid range is from 1 to 50. Default value is 50.

- `void SetITURP1812LocationPercentage(double location_percent)`
- `double GetITURP1812LocationPercentage()`

Sets/gets the percentage of locations for which the calculated basic transmission loss is not exceeded. Valid range is from 1 to 99. Default value is 50.

- `void SetITURP1812AverageRadioRefractivityLapseRate(double deltaN_Nunitskm)`
- `double GetITURP1812AverageRadioRefractivityLapseRate()`

Sets/gets the average radio-refractivity lapse-rate through the lowest 1 km of the atmosphere, in N-units/km. Alternately, *`deltaN_Nunitskm`* can be set to **`AUTOMATIC`**. In that case, the value is read from the ITU digital map file DN50.TXT using the path centre's geographical location. The file's location on disk must be specified using **`SetITUProprietaryDataDirectory`**. Default value is **`AUTOMATIC`**.

- `void SetITURP1812SeaLevelSurfaceRefractivity(double N0_Nunits)`
- `double GetITURP1812SeaLevelSurfaceRefractivity()`

Sets/gets the sea-level surface refractivity, in N-units. Alternately, *`N0_Nunits`* can be set to **`AUTOMATIC`**. In that case, the value is read from the ITU digital map file N050.TXT using the path centre's geographical location. The file's location on disk must be specified using **`SetITUProprietaryData-Directory`**. Default value is **`AUTOMATIC`**.

- `void SetITURP1812PredictionResolution(double resolution_meters)`
- `double GetITURP1812PredictionResolution()`

Sets/gets the prediction resolution, in meters. The prediction resolution is the width of the square area over which the variability applies. It must be greater than zero. Default value is 100 meters.

The prediction resolution parameter has no effect when the location percentage is set to 50. For more details, see the ITU-R P.1812-7 recommendation, sections 4.7 and 4.9 of Annex 1.

- `void SetITURP1812SurfaceProfileMethod(P1812SurfaceProfileMethod method)`
- `P1812SurfaceProfileMethod GetITURP1812SurfaceProfileMethod()`

Sets/gets the method to use in order to produce the surface profiles, as documented in the ITU-R P.1812-7 recommendation, sections 3.2.1 and 3.2.2 of Annex 1. `P1812SurfaceProfileMethod` is an enumeration type that may take one of the following values:

```
P1812_ADD_REPR_CLUTTER_HEIGHT  = 1
P1812_USE_SURFACE_ELEV_DATA    = 2
```

When **P1812_ADD_REPR_CLUTTER_HEIGHT** is selected, the surface profiles are produced by adding representative clutter heights to the terrain elevation values. This method uses both terrain elevation data (see **SetPrimaryTerrainElevDataSource**) and land cover data (see **SetPrimaryLandCoverData-Source**).

When **P1812_USE_SURFACE_ELEV_DATA** is selected, the surface profiles are produced by directly using surface elevation data, except at the transmitter and receiver locations where terrain elevation data is used instead. This method uses both terrain elevation data (see **SetPrimaryTerrainElevDataSource**) and surface elevation data (see **SetPrimarySurfaceElevDataSource**).

For both methods, spacing between points in the surface profiles is determined by the terrain elevation data sampling resolution (see **SetTerrainElevDataSamplingResolution**).

Default value for the surface profile method is **P1812_ADD_REPR_CLUTTER_HEIGHT**.

- `void` **SetITURP1812RepresentativeClutterHeight**(
  `P1812ClutterCategory` *clutterCategory*, `double` *reprHeight_meters*)
- `double` **GetITURP1812RepresentativeClutterHeight**(
  `P1812ClutterCategory` *clutterCategory*)

Sets/gets the representative height of the specified clutter category, in meters. `P1812ClutterCategory` is an enumeration type that may take one of the following values:

```
P1812_WATER_SEA           = 1
P1812_OPEN_RURAL          = 2
P1812_SUBURBAN            = 3
P1812_URBAN_TREES_FOREST  = 4
P1812_DENSE_URBAN         = 5
```

Default representative height values associated with each of the clutter categories are the following:

Clutter Category	Representative height (m)
P1812 WATER SEA	0
P1812 OPEN RURAL	0
P1812 SUBURBAN	10
P1812 URBAN TREES FOREST	15
P1812 DENSE URBAN	20

Clutter categories to apply along any profile from the transmitter to the receiver are determined by the specified land cover data source(s). Land cover data source(s) can be specified using the **SetPrimaryLandCoverDataSource** and **SetSecondaryLandCoverDataSource** methods. Land cover classes from the source(s) need to be mapped to the ITU-R P.1812 Rec. clutter categories. This can be achieved using the **SetLandCoverClassMapping** and **SetDefaultLandCoverClassMapping** methods. Default mappings may already exist depending on the selected land cover data source(s). Alternately, it is possible to directly map land cover classes to representative clutter heights instead of using the recommendation's clutter categories (see **SetITURP1812LandCoverMappingType**).

The number of land cover data samples to be used between the transmitter and receiver is determined by the terrain elevation data sampling resolution (see **SetTerrainElevDataSamplingResolution**) so that the number and location of samples will be identical for both the terrain elevation and land cover profiles.

When no land cover data source(s) is specified or that no land cover data is available at a particular location, the propagation model's implementation uses **P1812_OPEN_RURAL** as the default category.

Clutter categories and representative clutter heights are not used when the surface profile method is set to **P1812_USE_SURFACE_ELEV_DATA** (see **SetITURP1812SurfaceProfileMethod**).

For more details on clutter categories and representative clutter heights, see the ITU-R P.1812-7 recommendation, section 3.2 of Annex 1.

- `void SetITURP1812LandCoverMappingType(P1812LandCoverMappingType mappingType)`
- `P1812LandCoverMappingType GetITURP1812LandCoverMappingType()`

SetITURP1812LandCoverMappingType determines how the ITU-R P.1812 propagation model interprets the values that have been mapped to land cover classes. **P1812LandCoverMappingType** is an enumeration type that may take one of the following values:

```
P1812_MAP_TO_CLUTTER_CATEGORY     = 1
P1812_MAP_TO_REPR_CLUTTER_HEIGHT  = 2
```

When set to **P1812_MAP_TO_CLUTTER_CATEGORY**, mapped values are interpreted as clutter categories from **P1812ClutterCategory**. When set to **P1812_MAP_TO_REPR_CLUTTER_HEIGHT**, mapped values are interpreted as representative clutter heights in meters. Mapped values can be set or modified using the **SetLandCoverClassMapping** and **SetDefaultLandCoverClassMapping** methods.

The land cover mapping type is not used when the surface profile method is set to **P1812_USE_SURFACE_ELEV_DATA** (see **SetITURP1812SurfaceProfileMethod**).

Default value for the land cover mapping type is **P1812_MAP_TO_CLUTTER_CATEGORY**.

- `void SetITURP1812RadioClimaticZonesFile(const char* pathname)`
- `const char* GetITURP1812RadioClimaticZonesFile()`

Sets/gets the file to use as source for the ITU-R P.1812's radio climatic zones. The file must be of GeoTIFF format with a WGS 84 (EPSG: 4326) coordinate reference system, and with one of the following values at each pixel depending on the radio climatic zone:

Radio climatic zone	Pixel value
Coastal land	3
Inland	4
Sea	1

A file named rcz.tif is provided with CRC-COVLIB on a best effort basis to determine radio climatic zones as defined in the recommendation. Note that the ITU Digitized World Map (IDWM) product mentioned in the recommendation has not been used in any way to produce or validate the rcz.tif file. As an alternative to the rcz.tif file, a file named rcz_no_lakes.tif is also provided which omits defining a few of Canada's biggest lakes as a "Sea" zone.

pathname may either contain a relative or a full path. Default value for *pathname* is an empty string. When no file is specified, the "inland" zone type is assumed throughout the whole transmitter-receiver path.

For more details on radio climatic zones, see the ITU-R P.1812-7 recommendation, section 3.3 of Annex 1.

Additional implementation details for ITU-R P.1812

1) Exclusions

Building entry losses (section 4.8 of Annex 1) are excluded from the propagation model's implementation.

2) Troposcatter model

As advised on the ITU website (see https://www.itu.int/rec/R-REC-P.1812/en), this implementation of the ITU-R P.1812-7 propagation model reverts to using the troposcatter model from version 6 of the recommendation.

ITU-R P.452 propagation model methods

- `void SetITURP452TimePercentage(double time_percent)`
- `double GetITURP452TimePercentage()`

Sets/gets the time percentage for which the calculated basic transmission loss is not exceeded.

These methods (i.e. **`Set/GetITURP452TimePercentage`**) apply to all available versions (i.e. 17 & 18) of the propagation model.
Valid range is from 0.001 to 50. Default value is 50.

- `void SetITURP452PredictionType(P452PredictionType predictionType)`
- `P452PredictionType GetITURP452PredictionType()`

Sets/gets the prediction type (annual or “worst-month”). **`P452PredictionType`** is an enumeration type that may take one of the following values:

```
P452_AVERAGE_YEAR = 1
P452_WORST_MONTH  = 2
```

When the prediction type is set to **`P452_AVERAGE_YEAR`**, the time percentage value specified using **`SetITURP452TimePercentage`** is interpreted as the percentage of average year for which the transmission loss is not exceeded.

When the prediction type is set to **`P452_WORST_MONTH`**, the time percentage value specified using **`SetITURP452TimePercentage`** is interpreted as the percentage of worst-month for which the transmission loss is not exceeded. An annual equivalent time percentage of the worst-month time percentage is calculated as described in section 3.2.1 (Annex 1) of the ITU-R P.452-17/18 recommendation and used in further computations.

These methods (i.e. **`Set/GetITURP452PredictionType`**) apply to all available versions (i.e. 17 & 18) of the propagation model.
Default value is **`P452_AVERAGE_YEAR`**.

- `void SetITURP452AverageRadioRefractivityLapseRate(double deltaN_Nunitskm)`
- `double GetITURP452AverageRadioRefractivityLapseRate()`

Sets/gets the average radio-refractivity lapse-rate through the lowest 1 km of the atmosphere, in N-units/km. Alternately, *`deltaN_Nunitskm`* can be set to **`AUTOMATIC`**. In that case, the value is read from the ITU digital map file DN50.TXT using the path centre’s geographical location. The file’s location on disk must be specified using **`SetITUProprietaryDataDirectory`**.

These methods (i.e. **`Set/GetITURP452AverageRadioRefractivityLapseRate`**) apply to all available versions (i.e. 17 & 18) of the propagation model.
Default value is **`AUTOMATIC`**.

- `void` **`SetITURP452SeaLevelSurfaceRefractivity`**`(double` *`N0_Nunits`*`)`
- `double` **`GetITURP452SeaLevelSurfaceRefractivity`**`()`

Sets/gets the sea-level surface refractivity, in N-units. Alternately, *`N0_Nunits`* can be set to **`AUTOMATIC`**. In that case, the value is read from the ITU digital map file N050.TXT using the path centre's geographical location. The file's location on disk must be specified using **`SetITUProprietaryData-Directory`**.

These methods (i.e. **`Set/GetITURP452SeaLevelSurfaceRefractivity`**) apply to all available versions (i.e. 17 & 18) of the propagation model.
Default value is **`AUTOMATIC`**.

- `void` **`SetITURP452AirTemperature`**`(double` *`temperature_C`*`)`
- `double` **`GetITURP452AirTemperature`**`()`

Sets/gets the air temperature, in degree Celsius. Alternately, *`temperature_C`* can be set to **`AUTOMATIC`**. In that case, the annual mean surface temperature from recommendation ITU-R P.1510-1 is used. The value is read from the ITU digital map file T_Annual.TXT using the path centre's geographical location. The file's location on disk must be specified using **`SetITUProprietaryDataDirectory`**.

These methods (i.e. **`Set/GetITURP452AirTemperature`**) apply to all available versions (i.e. 17 & 18) of the propagation model.
Default value is **`AUTOMATIC`**.

- `void` **`SetITURP452AirPressure`**`(double` *`pressure_hPa`*`)`
- `double` **`GetITURP452AirPressure`**`()`

Sets/gets the air pressure (also know as atmospheric pressure), in hectopascals (hPa). Alternately, *`pressure_hPa`* can be set to **`AUTOMATIC`**. In that case, the mean annual global reference atmosphere pressure from recommendation ITU-R P.835-6 (Annex 1, Section 1.1) at the path centre's terrain height is used.

These methods (i.e. **`Set/GetITURP452AirPressure`**) apply to all available versions (i.e. 17 & 18) of the propagation model.
Default value is **`AUTOMATIC`**.

- `void` **`SetITURP452RadioClimaticZonesFile`**`(const char*` *`pathname`*`)`
- `const char*` **`GetITURP452RadioClimaticZonesFile`**`()`

Sets/gets the file to use as source for the ITU-R P.452's radio climatic zone data. The file must be of GeoTIFF format with a WGS 84 (EPSG: 4326) coordinate reference system, and with one of the following values at each pixel depending on the radio climatic zone:

Radio climatic zone	Pixel value
Coastal land	3
Inland	4
Sea	1

A file named rcz.tif is provided with CRC-COVLIB on a best effort basis to determine radio climatic zones as defined in the recommendation. Note that the ITU Digitized World Map (IDWM) product mentioned in the recommendation has not been used in any way to produce or validate the rcz.tif file. As an alternative to the rcz.tif file, a file named rcz_no_lakes.tif is also provided which omits defining a few of Canada's biggest lakes as a "Sea" zone.

pathname may either contain a relative or a full path. Default value for *pathname* is an empty string. When no file is specified, the "inland" zone type is assumed throughout the whole transmitter-receiver path.

These methods (i.e. **Set/GetITURP452RadioClimaticZonesFile**) apply to all available versions (i.e. 17 & 18) of the propagation model.

For more details on radio climatic zones, see the ITU-R P.452-17/18 recommendation, section 3.2.1 of Annex 1.

- **void SetITURP452HeightGainModelClutterValue(**
  P452HeightGainModelClutterCategory *clutterCategory*,
  P452HeightGainModelClutterParam *nominalParam*, **double** *nominalValue*)
- **double GetITURP452HeightGainModelClutterValue(**
  P452HeightGainModelClutterCategory *clutterCategory*,
  P452HeightGainModelClutterParam *nominalParam*)

Sets/gets the nominal height (in meters) or nominal distance (in kilometres) of the specified height-gain model clutter category. **P452HeightGainModelClutterCategory** is an enumeration type that may take one of the following values:

P452_HGM_HIGH_CROP_FIELDS	= 13
P452_HGM_PARK_LAND	= 19
P452_HGM_IRREGULARLY_SPACED_SPARSE_TREES	= 21
P452_HGM_ORCHARD_REGULARLY_SPACED	= 22
P452_HGM_SPARSE_HOUSES	= 31
P452_HGM_VILLAGE_CENTRE	= 32
P452_HGM_DECIDUOUS_TREES_IRREGULARLY_SPACED	= 23
P452_HGM_DECIDUOUS_TREES_REGULARLY_SPACED	= 24
P452_HGM_MIXED_TREE_FOREST	= 27
P452_HGM_CONIFEROUS_TREES_IRREGULARLY_SPACED	= 25
P452_HGM_CONIFEROUS_TREES_REGULARLY_SPACED	= 26
P452_HGM_TROPICAL_RAIN_FOREST	= 28
P452_HGM_SUBURBAN	= 33
P452_HGM_DENSE_SUBURBAN	= 34
P452_HGM_URBAN	= 35
P452_HGM_DENSE_URBAN	= 36
P452_HGM_HIGH_RISE_URBAN	= 38
P452_HGM_INDUSTRIAL_ZONE	= 37

```
P452_HGM_OTHER                              = 90
P452_HGM_CUSTOM_AT_TRANSMITTER              = 200
P452_HGM_CUSTOM_AT_RECEIVER                 = 201
```

`P452HeightGainModelClutterParam` is an enumeration type that may take one of the following values:

```
P452_NOMINAL_HEIGHT_M    = 1
P452_NOMINAL_DISTANCE_KM = 2
```

nominalValue should either be set to the new nominal height (in meters) or nominal distance (in kilometres) depending on the specified value for *nominalParam*. Default nominal values are listed in the table below:

Clutter Category	Nominal Height (m)	Nominal Distance (km)
P452_HGM_HIGH_CROP_FIELDS	4.0	0.1
P452_HGM_PARK_LAND	4.0	0.1
P452_HGM_IRREGULARLY_SPACED_SPARSE_TREES	4.0	0.1
P452_HGM_ORCHARD_REGULARLY_SPACED	4.0	0.1
P452_HGM_SPARSE_HOUSES	4.0	0.1
P452_HGM_VILLAGE_CENTRE	5.0	0.07
P452_HGM_DECIDUOUS_TREES_IRREGULARLY_SPACED	15.0	0.05
P452_HGM_DECIDUOUS_TREES_REGULARLY_SPACED	15.0	0.05
P452_HGM_MIXED_TREE_FOREST	15.0	0.05
P452_HGM_CONIFEROUS_TREES_IRREGULARLY_SPACED	20.0	0.05
P452_HGM_CONIFEROUS_TREES_REGULARLY_SPACED	20.0	0.05
P452_HGM_TROPICAL_RAIN_FOREST	20.0	0.03
P452_HGM_SUBURBAN	9.0	0.025
P452_HGM_DENSE_SUBURBAN	12.0	0.02
P452_HGM_URBAN	20.0	0.02
P452_HGM_DENSE_URBAN	25.0	0.02
P452_HGM_HIGH_RISE_URBAN	35.0	0.02
P452_HGM_INDUSTRIAL_ZONE	20.0	0.05
P452_HGM_OTHER	0.0	0.0
P452_HGM_CUSTOM_AT_TRANSMITTER	0.0	0.0
P452_HGM_CUSTOM_AT_RECEIVER	0.0	0.0

Whether a clutter category and which clutter category is applied at either end of the transmission path in order to calculate additional clutter losses depend on the specified value to the **SetITURP452HeightGainModelMode** method.

These methods (i.e. **Set/GetITURP452HeightGainModelClutterValue**) only apply to version 17 of the propagation model.

For more details on the height-gain model clutter categories and nominal heights and distances, see the ITU-R P.452-17 recommendation, section 4.5 of Annex 1.

- `void SetITURP452HeightGainModelMode(Terminal terminal, P452HeightGainModelMode mode)`
- `P452HeightGainModelMode GetITURP452HeightGainModelMode(Terminal terminal)`

Sets/gets the height-gain model mode at either end of the transmission path. **Terminal** is an enumeration type that may take one of the following values:

```
TRANSMITTER = 1
RECEIVER    = 2
```

P452HeightGainModelMode is an enumeration type that may take one of the following values:

```
P452_NO_SHIELDING            = 1
P452_USE_CUSTOM_AT_CATEGORY  = 2
P452_USE_CLUTTER_PROFILE     = 3
P452_USE_CLUTTER_AT_ENDPOINT = 4
```

P452_NO_SHIELDING
No clutter losses will be added.

P452_USE_CUSTOM_AT_CATEGORY
When *mode* is set to **P452_P452_USE_CUSTOM_AT_CATEGORY** for the **TRANSMITTER**'s end of the path, the nominal height and distance of the **P452_HGM_CUSTOM_AT_TRANSMITTER** clutter category will be used for the clutter losses calculation at the transmitter site. Likewise, when *mode* is set to **P452_USE_CUSTOM_AT_CATEGORY** for the **RECEIVER**'s end of the path, the nominal height and distance of the **P452_HGM_CUSTOM_AT_RECEIVER** clutter category will be used for the clutter losses calculation at the receiver site.

P452_USE_CLUTTER_PROFILE
A clutter category profile between the transmitter and the receiver will be used. The clutter category with the highest nominal height value to be found along this profile within 100 meters of the transmitter or receiver will be retained. The nominal height and distance values of this retained clutter category will then be used in the clutter losses calculation.

P452_USE_CLUTTER_AT_ENDPOINT
The nominal height and distance values of the clutter category found at the transmitter or receiver location will be used in the clutter losses calculation.

Height-gain model clutter categories at the transmitter or receiver locations or along any profile between the two are determined by the specified land cover data source(s). Land cover data source(s) can be specified using the **SetPrimaryLandCoverDataSource** and **SetSecondaryLandCoverData-Source** methods. Land cover classes from the source(s) need to be mapped to the ITU-R P.452-17 height-gain model clutter categories. This can be achieved using the **SetLandCoverClassMapping** and **SetDefaultLandCoverClassMapping** methods. Default mappings may already exist depending on the selected land cover data source(s). The number of land cover data samples to be used for a profile between the transmitter and receiver is determined by the terrain elevation data sampling resolution (**SetTerrainElevDataSamplingResolution** method) so that the number and location of samples will be identical for both the terrain elevation and land cover profiles. When no land cover data source(s) is specified or that no land cover data is available at a particular location, the propagation model's implementation uses **P452_HGM_OTHER** as the default category.

These methods (i.e. **Set/GetITURP452HeightGainModelMode**) only apply to version 17 of the propagation model.
Default value is **P452_USE_CLUTTER_AT_ENDPOINT** at both ends of the transmission path.

- `void SetITURP452RepresentativeClutterHeight(P452ClutterCategory clutterCategory, double reprHeight_meters)`
- `double GetITURP452RepresentativeClutterHeight(P452ClutterCategory clutterCategory)`

Sets/gets the representative height of the specified clutter category, in meters. **P452ClutterCategory** is an enumeration type that may take one of the following values:

```
P452_WATER_SEA           = 1
P452_OPEN_RURAL          = 2
P452_SUBURBAN            = 3
P452_URBAN_TREES_FOREST  = 4
P452_DENSE_URBAN         = 5
```

Default representative height values associated with each of the clutter categories are the following:

Clutter Category	Representative height (m)
P452_WATER_SEA	0
P452_OPEN_RURAL	0
P452_SUBURBAN	10
P452_URBAN_TREES_FOREST	15
P452_DENSE_URBAN	20

Clutter categories to apply along any profile from the transmitter to the receiver are determined by the specified land cover data source(s). Land cover data source(s) can be specified using the **SetPrimaryLandCoverDataSource** and **SetSecondaryLandCoverDataSource** methods. Land cover classes from the source(s) need to be mapped to the ITU-R P.452-18 clutter categories. This can be achieved using the **SetLandCoverClassMapping** and **SetDefaultLandCoverClassMapping** methods. Default mappings may already exist depending on the selected land cover data source(s). Alternately, it is possible to directly map land cover classes to representative clutter heights instead of using the recommendation's clutter categories (see **SetITURP452LandCoverMappingType**).

The number of land cover data samples to be used between the transmitter and receiver is determined by the terrain elevation data sampling resolution (**SetTerrainElevDataSamplingResolution** method) so that the number and location of samples will be identical for both the elevation and land cover profiles.

When no land cover data source(s) is specified or that no land cover data is available at a particular location, the propagation model's implementation uses **P452_OPEN_RURAL** as the default category.

These methods (i.e. **Set/GetITURP452RepresentativeClutterHeight**) only apply to version 18 of the propagation model.

For more details on clutter categories and representative clutter heights, see the ITU-R P.452-18 recommendation, section 3.2.1 of Annex 1.

- `void SetITURP452LandCoverMappingType(P452LandCoverMappingType mappingType)`
- `P452LandCoverMappingType GetITURP452LandCoverMappingType()`

The **SetITURP452LandCoverMappingType** determines how the ITU-R P.452-18 propagation model interprets the values that have been mapped to land cover classes. `P452LandCoverMappingType` is an enumeration type that may take one of the following values:

```
P452_MAP_TO_CLUTTER_CATEGORY     = 1
P452_MAP_TO_REPR_CLUTTER_HEIGHT  = 2
```

When set to **P452_MAP_TO_CLUTTER_CATEGORY**, mapped values are interpreted as clutter categories from `P452ClutterCategory`. When set to **P452_MAP_TO_REPR_CLUTTER_HEIGHT**, mapped values are interpreted as representative clutter heights in meters. Mapped values can be set or modified using the **SetLandCoverClassMapping** and **SetDefaultLandCoverClassMapping** methods.

These methods (i.e. **Set/GetITURP452LandCoverMappingType**) only apply to version 18 of the propagation model.
Default value is **P452_MAP_TO_CLUTTER_CATEGORY**.

- `void SetITURP452SurfaceProfileMethod(P452SurfaceProfileMethod method)`
- `P452SurfaceProfileMethod GetITURP452SurfaceProfileMethod()`

In an experimental context, the terrain profile described in ITU-R P.452-18 (step 4 of section 3.2.1 from Annex 1) may be alternately produced by directly using surface elevation data, in a similar manner to what is proposed in ITU-R P.1812-7 (section 3.2.2 of Annex 1). `P452SurfaceProfileMethod` is an enumeration type that may take one of the following values:

```
P452_ADD_REPR_CLUTTER_HEIGHT                 = 1
P452_EXPERIMENTAL_USE_OF_SURFACE_ELEV_DATA  = 2
```

When **P452_ADD_REPR_CLUTTER_HEIGHT** is selected, the required profile is produced by adding representative clutter heights to the terrain elevation values. This method uses both terrain elevation data (see **SetPrimaryTerrainElevDataSource**) and land cover data (see **SetPrimaryLandCoverData-Source**).

When **P452_EXPERIMENTAL_USE_OF_SURFACE_ELEV_DATA** is selected, the required profile is produced by directly using surface elevation data, except within 50 m of the transmitter and receiver locations where terrain elevation data is used instead. This method uses both terrain elevation data (see **SetPrimaryTerrainElevDataSource**) and surface elevation data (see **SetPrimarySurfaceElevData-Source**). Note that this method for producing profiles is not currently part of the ITU-R P.452-18 recommendation.

For both methods, spacing between points in the surface profiles is determined by the terrain elevation data sampling resolution (see **`SetTerrainElevDataSamplingResolution`**).

These methods (i.e. **`Set/GetITURP452SurfaceProfileMethod`**) only apply to version 18 of the propagation model.
Default value is **`P452_ADD_REPR_CLUTTER_HEIGHT`**.

Additional implementation details for ITU-R P.452

1) Interfering station VS interfered-with station

   In the context of the ITU-R P.452 propagation model, CRC-COVLIB's transmitter holds the role of the interferer while the receiver is the terminal being interfered-with.

2) Inclusions / exclusions

   CRC-COVLIB's implementation of ITU-R P.452 corresponds to the clear-air interference prediction of section 3 and 4 (Annex 1) of the recommendation. It does not include the hydrometeor-scatter interference prediction of section 5.

3) Antenna gains

   Values for G_t and G_r (tropospheric scatter section of the recommendation) are obtained from the specified antenna parameters for the simulation. If no antenna parameters were specified, default values of 0 dBi are utilized for both G_t and G_r.

Extended-Hata urban propagation model methods

- `void SetEHataClutterEnvironment(EHataClutterEnvironment clutterEnvironment)`
- `EHataClutterEnvironment GetEHataClutterEnvironment()`

Sets/gets the clutter environment. `EHataClutterEnvironment` is an enumeration type that may take one of the following values:

```
EHATA_URBAN    = 24
EHATA_SUBURBAN = 22
EHATA_RURAL    = 20
```

Default value is `EHATA_URBAN`.

- `void SetEHataReliabilityPercentage(double percent)`
- `double GetEHataReliabilityPercentage()`

Sets/gets the percent not exceeded of the signal. *`percent`* must be greater than 0 and smaller than 100. Default value is 50.

Additional resources

Additional resources regarding the Extended-Hata urban propagation model are available at:

https://github.com/NTIA/ehata

ITU-R P.2108 clutter loss model methods

- `void SetITURP2108TerrestrialStatModelActiveState(bool active)`
- `double GetITURP2108TerrestrialStatModelActiveState()`

Sets/gets the active state of the statistical clutter loss model for terrestrial paths from the ITU-R P.2108-1 recommendation (Section 3.2).

When *active* is set to **true**, additional clutter losses are added to path losses when generating results using the **GenerateReceptionPointResult**, **GenerateReceptionPointDetailedResult**, **Generate-ReceptionAreaResults**, **GenerateProfileReceptionPointResult** and **ExportProfilesToCsvFile** methods. The clutter loss value to be added is calculated using the transmitter's frequency, the distance between the transmitter and the reception point and the location percentage specified using **SetITURP2108TerrestrialStatModelLocation-Percentage**.

This clutter loss model is valid for frequencies going from 0.5 to 67 GHz and for path lengths of at least 0.25 km. When the model is used outside of those validity ranges, no clutter loss is added.

Note that the ITU recommendation also mentions that this model applies to urban and suburban environments where terminal heights are well below the clutter heights, and that it should not be used with propagation models that inherently account for clutter losses over the entire path. The observance of those latter conditions falls upon CRC-COVLIB's user responsibility.

Default value for the active state is **false**.

- `void SetITURP2108TerrestrialStatModelLocationPercentage(double location_percent)`
- `double GetITURP2108TerrestrialStatModelLocationPercentage()`

Sets/gets the location percentage for which the clutter loss for terrestrial paths is not exceeded. Valid range is from 0.000001 to 99.999999. Default value is 50.

- `double GetITURP2108TerrestrialStatModelLoss(double frequency_GHz,`
  `double distance_km)`

Gets the calculated clutter loss for terrestrial paths, in dB, given the specified frequency (GHz), distance (km) and location percentage specified using **SetITURP2108TerrestrialStatModelLocation-Percentage**. Valid range for *frequency_GHz* is from 0.5 to 67 GHz and 0.25 km and up for *distance_km*.

ITU-R P.2109 building entry loss model methods

- `void SetITURP2109ActiveState(bool active)`
- `double GetITURP2109ActiveState()`

Sets/gets the active state of the building entry loss model from the ITU-R P.2109-2 recommendation.

When *active* is set to **true**, additional building entry losses are added to path losses when generating results using the **GenerateReceptionPointResult**, **GenerateReceptionPointDetailedResult**, **GenerateReceptionAreaResults**, **GenerateProfileReceptionPointResult** and **ExportProfilesTo-CsvFile** methods. The building entry loss value depends on four parameters: the transmitter's frequency, the probability for which the entry loss is not exceeded (specified via **SetITURP2109Probability**), the building type (specified via **SetITURP2109DefaultBuildingType**) and the elevation angle of the path at the building façade. The elevation angle is calculated from the transmitter and receiver antenna heights and from the terrain elevation profile. The calculated building entry loss is added once at every reception point involved, assuming either that the transmitter is outside and the receiver inside, or that the transmitter is inside and the receiver outside, and that the signal traverses only one façade wall.

This clutter loss model is valid for frequencies going from 0.08 to 100 GHz. When the model is used outside of this validity range, no building entry loss is added.

Default value for the active state is **false**.

- `void SetITURP2109Probability(double probability_percent)`
- `double GetITURP2109Probability()`

Sets/gets the probability for which the building entry loss is not exceeded. Valid range is from 0.000001 to 99.999999. Default value is 50.

- `void SetITURP2109DefaultBuildingType(P2109BuildingType buildingType)`
- `P2109BuildingType GetITURP2109DefaultBuildingType()`

Sets/gets the building type to use when calculating entry losses. **P2109BuildingType** is an enumeration type that may take one of the following values:

```
P2109_TRADITIONAL         = 1
P2109_THERMALLY_EFFICIENT = 2
```

Default value is **P2109_TRADITIONAL**.

- `double GetITURP2109BuildingEntryLoss(double frequency_GHz,`
  `double elevAngle_degrees)`

Gets the calculated building entry loss, in dB, given the specified frequency (GHz) and the elevation angle of the path at the building façade (degrees from the horizontal). The calculation also takes into account the probability and building type specified via **SetITURP2109Probability** and **SetITURP2109-DefaultBuildingType**. Valid range for *frequency_GHz* is from 0.08 to 100 GHz and -90 to 90 degrees inclusively for *elevAngle_degrees*.

ITU-R P.676 gaseous attenuation model for terrestrial paths methods

- `void SetITURP676TerrPathGaseousAttenuationActiveState(bool active, double atmPressure_hPa=AUTOMATIC, double temperature_C=AUTOMATIC, double waterVapourDensity_gm3=AUTOMATIC)`
- `bool GetITURP676TerrPathGaseousAttenuationActiveState()`

Sets/gets the active state of the gaseous attenuation for terrestrial paths model from the ITU-R P.676-13 recommendation (Annex 1, Sections 1 & 2.1). This gaseous attenuation model is valid for any frequency up to 1000 GHz. When the model is used outside of this validity range, no gaseous attenuation is added to path losses.

When *active* is set to **true**, attenuation losses due to atmospheric gases are added to path losses when generating results using the **GenerateReceptionPointResult**, **GenerateReceptionPointDetailed-Result**, **GenerateReceptionAreaResults**, **GenerateProfileReceptionPointResult** and **Export-ProfilesToCsvFile** methods. The gaseous attenuation loss value depends on five parameters: the transmitter's frequency (from **SetTransmitterFrequency**), the path length between transmitter and receiver (always automatically calculated), the atmospheric pressure (hPa), the temperature (°C) and the water vapour density (g/m^3). Each of those last three parameters may either be set to a specific value or to the **AUTOMATIC** constant.

When *atmPressure_hPa* is set to **AUTOMATIC**, the mean annual global reference atmosphere pressure at the terrain height of the mid-point between the receiver and the transmitter is used (ITU-R P.835-6, Annex 1, Section 1.1).

When *temperature_C* is set to **AUTOMATIC**, the annual mean surface temperature from recommendation ITU-R P.1510-1 is used. The value is read from the ITU digital map file T_Annual.TXT at mid-point between the receiver and the transmitter. The file's location on disk must be specified using **SetITUProprietaryDataDirectory**.

When *waterVapourDensity_gm3* is set to **AUTOMATIC**, the surface water-vapour density under non-rain conditions, exceeded for 50% of an average year, at mid-point between the receiver and the transmitter is used (ITU-R P.2001-5). This value is read from the ITU digital map file Surfwv_50_fixed.txt. The file's location on disk must be specified using **SetITUProprietaryDataDirectory**.

Some propagation models like ITU-R P.452 already take gaseous attenuation into account. Do not activate the gaseous attenuation model referred to here with such models, otherwise the attenuation will be added twice.

Default value for the active state is **false**.

- `double GetITURP676GaseousAttenuation(double frequency_GHz, double atmPressure_hPa=1013.25, double temperature_C=15, double waterVapourDensity_gm3=7.5)`

Gets the gaseous attenuation, in dB/km, according to the ITU-R P.676-13 recommendation (Annex 1, Section 1). A frequency in GHz, up to 1000 GHz, must be specified. Values for atmospheric pressure (hPa), temperature (°C) and water-vapour density (g/m^3) may also be specified.

ITU digial maps methods

- **double GetITUDigitalMapValue(ITUDigitalMap *map*, double *latitude_degrees*, double *longitude_degrees*)**

Gets the interpolated value from the specified ITU digital map at the specified geographical location. The latitude value must be between -90 and +90 degrees while the longitude value must be between -180 and +180 degrees.

The location on disk of the ITU digital map files must be specified using the **SetITUProprietaryDataDirectory** function from the `Crc::Covlib` namespace. If CRC-COVLIB cannot find or read the digital map files, **GetITUDigitalMapValue** returns a default value (see below). This default value is also used internally whenever a value from a missing digital map file is requested.

`ITUDigitalMap` is an enumeration type that may take one of the following values:

```
ITU_MAP_DN50      = 1
ITU_MAP_N050      = 2
ITU_MAP_T_ANNUAL  = 3
ITU_MAP_SURFWV_50 = 4
```

ITU_MAP_DN50
Median annual radio-refractivity lapse-rate through the lowest 1 km of the atmosphere (N-units/km).
Default value is 45 N-units/km.

ITU_MAP_N050
Median annual sea-level surface refractivity (N-units).
Default value is 325 N-units.

ITU_MAP_T_ANNUAL
Annual mean surface temperature at 2 meters above the surface of the Earth (Kelvins).
Default value is 288.15 K.

ITU_MAP_SURFWV_50
Surface water-vapour density under non-rain conditions, exceeded for 50% of an average year (g/m^3).
Default value is 7.5 g/m^3.

Terrain elevation data methods

- `void SetPrimaryTerrainElevDataSource(TerrainElevDataSource terrainElevSource)`
- `TerrainElevDataSource GetPrimaryTerrainElevDataSource()`
- `void SetSecondaryTerrainElevDataSource(TerrainElevDataSource terrainElevSource)`
- `TerrainElevDataSource GetSecondaryTerrainElevDataSource()`
- `void SetTertiaryTerrainElevDataSource(TerrainElevDataSource terrainElevSource)`
- `TerrainElevDataSource GetTertiaryTerrainElevDataSource()`

Sets/gets the source(s) for the terrain elevation data (i.e. ground level). The primary source is the first source CRC-COVLIB tries to get the terrain elevation data from. If terrain elevation data cannot be obtained from the primary source for a given location, CRC-COVLIB will try to get it from the secondary source. Similarly, when no terrain elevation data can be obtained from both the primary and the secondary sources, CRC-COVLIB will use the tertiary source. Where no terrain elevation data can be obtained at all, a terrain elevation value of 0 meter is provided to the propagation model. Note that CRC-COVLIB does not make any correction for using different vertical datums.

`TerrainElevDataSource` is an enumeration type that may take one of the following values:

```
TERR_ELEV_NONE             = 0
TERR_ELEV_SRTM             = 1
TERR_ELEV_CUSTOM           = 2
TERR_ELEV_NRCAN_CDEM       = 3
TERR_ELEV_NRCAN_HRDEM_DTM  = 4
TERR_ELEV_GEOTIFF          = 5
TERR_ELEV_NRCAN_MRDEM_DTM  = 6
```

`TERR_ELEV_NONE`
No terrain elevation data source.

`TERR_ELEV_SRTM` (NASA Shuttle Radar Topography Mission)
Please note that SRTM files usually contain surface elevation data (i.e. ground level + clutter height). However they may still be used “as if” they were terrain elevation files if no better data source is available at the location of interest.
Files must be unzipped and their location on disk should be specified using the **`SetTerrainElevDataSourceDirectory`** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible files (the *.hgt extension is usually used for 1 and 3 arcseconds data and the *.dem extension for 30 arcseconds data). File names specifying start location in latitude and longitude must be preserved. Note that data files originating from the Shuttle Radar Topography Mission (SRTM) may also be made available in the GeoTIFF (*.tif) format. In that case, `TERR_ELEV_GEOTIFF` must be specified as the source instead of `TERR_ELEV_SRTM`.

`TERR_ELEV_CUSTOM`
Uses terrain elevation data submitted trough calls to the **`AddCustomTerrainElevData`** method.

`TERR_ELEV_NRCAN_CDEM` (Natural Resources Canada / Canadian Digital Elevation Model)
Compatible files for this source can be obtained at:
https://ftp.maps.canada.ca/pub/nrcan_rncan/elevation/cdem_mnec/

https://download-telecharger.services.geo.ca/pub/nrcan_rncan/elevation/cdem_mnec/
Files should be unzipped and their location on disk should be specified using the **SetTerrainElevDataSourceDirectory** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible *.tif files. Other file types (*.pdf, *.xml) may be present but are not required nor used. ***Note: CDEM is now a legacy product, a better option is to use the Medium Resolution Digital Elevation Model (MRDEM) product (see* TERR_ELEV_NRCAN_MRDEM_DTM *below).***

TERR_ELEV_NRCAN_HRDEM_DTM (Natural Resources Canada / High Resolution Digital Elevation Model / Digital Terrain Model)
Compatible files for this source can be obtained at:
https://ftp.maps.canada.ca/pub/elevation/dem_mne/highresolution_hauteresolution/dtm_mnt/
https://download-telecharger.services.geo.ca/pub/elevation/dem_mne/highresolution_hauteresolution/dtm_mnt/
Location of files on disk should be specified using the **SetTerrainElevDataSourceDirectory** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible *.tif files. Note that the files that may be used are those with names starting with 'dtm_' (example: dtm_1m_utm18_w_1_102.tif).
The following tool may also be used to search for and download HRDEM files:
https://search.open.canada.ca/openmap/957782bf-847c-4644-a757-e383c0057995
Note that the ArticDEM mosaic files (in the polar stereographic North coordinate system) are not supported.
For additional details on this product, please see:
https://open.canada.ca/data/en/dataset/957782bf-847c-4644-a757-e383c0057995
https://open.canada.ca/data/en/dataset/0fe65119-e96e-4a57-8bfe-9d9245fba06b

TERR_ELEV_GEOTIFF
Use provided GeoTIFF files as terrain elevation source. Location of files on disk should be specified using the **SetTerrainElevDataSourceDirectory** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible *.tif files. Only a few coordinate reference systems are currently supported, namely "WGS 84" (EPSG:4326), "NAD83" (EPSG:4269), "NAD83(CSRS)" (EPSG:4617), "NAD83(CSRS98)" (EPSG:4140), "WGS 84 / UTM zone [num]", "NAD83 / UTM zone [num]", "NAD83(CSRS) / UTM zone [num]" and "NAD83(CSRS) / Canada Atlas Lambert" (EPSG:3979).

TERR_ELEV_NRCAN_MRDEM_DTM (Natural Resources Canada / Medium Resolution Digital Elevation Model / Digital Terrain Model)
This product consists in a large cloud optimized GeoTIFF file (56.1 GB) available at:
https://datacube-prod-data-public.s3.ca-central-1.amazonaws.com/store/elevation/mrdem/mrdem-30/mrdem-30-dtm.tif
CRC-COVLIB can either use the whole file as is or extracts from it. In both cases, the file(s)' location on disk should be specified using the **SetTerrainElevDataSourceDirectory** method. For additional details on this product, please see:
https://open.canada.ca/data/en/dataset/18752265-bda3-498c-a4ba-9dfe68cb98da

Note: CRC-COVLIB's python package contains functions to facilitate the download of extracts from the CDEM, HRDEM and MRDEM products. They are available under the crc_covlib.helper.datacube_canada module.

Default value for the primary terrain elevation data source is **ELEV_NONE**.
Default value for the secondary terrain elevation data source is **ELEV_NONE**.

Default value for the tertiary terrain elevation data source is **ELEV_NONE**.

- `void SetTerrainElevDataSourceDirectory(TerrainElevDataSource terrainElevSource, const char* directory, bool useIndexFile=false, bool overwriteIndexFile=false)`
- `const char* GetTerrainElevDataSourceDirectory(TerrainElevDataSource terrainElevSource)`

Sets/gets the directory associated with the specified terrain elevation data source.

For terrain elevation data sources **TERR_ELEV_NRCAN_CDEM**, **TERR_ELEV_NRCAN_HRDEM_DTM** and **TERR_ELEV_GEOTIFF**, an index file can be used by setting the *useIndexFile* parameter to **true**. This may be useful for directories containing a great number of files in order to prevent CRC-COVLIB from reading the whole directory and sub-directories content each time the **SetTerrainElevDataSourceDirectory** method is called. This option creates an index file the first time the directory content is read and subsequently read that single file only as information source. Note that when directories or files are added, removed or modified within a directory using an index file, that index file needs to be regenerated in order for CRC-COVLIB to see the changes. This can be achieved by manually deleting the index file or by setting the *overwriteIndexFile* parameter to **true**. For terrain elevation data sources other than those mentioned above, the *useIndexFile* and *overwriteIndexFile* parameters have no effect.

Default value for the directory of any terrain elevation data source is an empty string.

- `void SetTerrainElevDataSourceSamplingMethod(TerrainElevDataSource terrainElevSource, SamplingMethod samplingMethod)`
- `SamplingMethod GetTerrainElevDataSourceSamplingMethod(TerrainElevDataSource terrainElevSource)`

Sets/gets the sampling method for the specified terrain elevation data source. `SamplingMethod` is an enumeration type that may take one of the following values:

```
NEAREST_NEIGHBOR       = 0
BILINEAR_INTERPOLATION = 1
```

The sampling method applies when reading terrain elevation data files. The nearest neighbor method requires reading one value from file while the bilinear interpolation method requires four. For this reason the nearest neighbor method is normally faster. Default value is **BILINEAR_INTERPOLATION** for all terrain elevation data sources.

- `void SetTerrainElevDataSamplingResolution(double samplingResolution_meters)`
- `double GetTerrainElevDataSamplingResolution()`

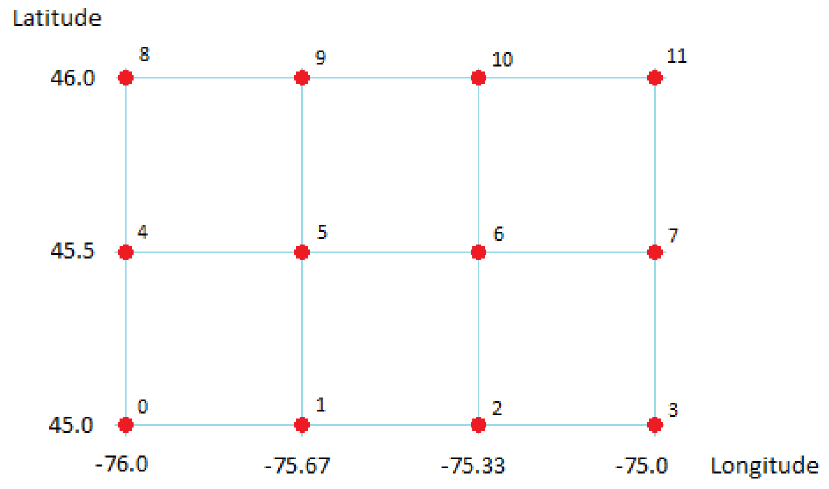
Sets/gets the terrain elevation data sampling resolution. Whenever a simulation result (field strength, path loss, transmission loss or received power) needs to be calculated, a terrain elevation profile starting from the transmitter to the reception point may need to be provided to the propagation model. CRC-COVLIB generates the terrain elevation profiles from the specified terrain elevation data source(s) at the specified sampling resolution. For example, when the sampling resolution is set to 30 meters, the terrain elevation profiles will contain a terrain elevation data point every 30 meters or so. The terrain elevation data sampling resolution also applies to any land cover or surface elevation profile required by the selected propagation model. This is to ensure that the number and location of samples will be identical throughout all profiles.

Default value for the terrain elevation data sampling resolution is 100 meters.

-
  ```
  bool AddCustomTerrainElevData(
    double lowerLeftCornerLat_degrees, double lowerLeftCornerLon_degrees,
    double upperRightCornerLat_degrees, double upperRightCornerLon_degrees,
    int numHorizSamples, int numVertSamples, const float* terrainElevData_meters,
    bool defineNoDataValue=false, float noDataValue=0)
  ```

Adds custom terrain elevation data. *terrainElevData_meters* should point to a contiguous list of *numHorizSamples* times *numVertSamples* terrain elevation values. Terrain elevation samples are assumed to be equally spaced (i.e. spacing being measured in degrees) along latitude and longitude lines. The first terrain elevation value pointed to by *terrainElevData_meters* is assumed to be geographically located at the specified lower left corner position. The list should then continue first horizontally and then up, up to the specified upper right corner position. For example:

```
AddCustomTerrainElevData(45.0, -76.0, 46.0, -75.0, 4, 3, arrayOf12Floats);
```



If parts of the specified rectangular area do not have valid terrain elevation information, this may be specified by using a “no data” value within *terrainElevData_meters*. If a specific value within *terrainElevData_meters* must be interpreted as “no data”, this should be indicated by setting the *defineNoDataValue* parameter to **true** and the *noDataValue* parameter to this value. When *defineNoDataValue* is set to **false**, *noDataValue* is ignored.

AddCustomTerrainElevData allocates new memory within the **Simulation** object and copies the data pointed to by *terrainElevData_meters*. The data pointed to by *terrainElevData_meters* is therefore not required to be preserved after the call to **`AddCustomTerrainElevData`**. The copied data is preserved until the `Simulation` object is destroyed (via **`Release`**) or until a call to **`ClearCustom-TerrainElevData`** on the same `Simulation` object..

Returns `false` on failure to allocate sufficient memory, `true` otherwise.

- `void` **`ClearCustomTerrainElevData()`**

Removes all custom terrain elevation data previously added through **`AddCustomTerrainElevData`** and releases corresponding memory.

- `double` **`GetTerrainElevation(`**`double` *`latitude_degrees`*, `double` *`longitude_degrees`*, `double` *`noDataValue`*`=0)`

Gets the terrain elevation in meters at the specified location using the terrain elevation data source(s) that were specified using the **`SetPrimaryTerrainElevDataSource`**, **`SetSecondaryTerrainElevData-Source`** and **`SetTertirayTerrainElevDataSource`** methods. If no terrain elevation data can be obtained at the specified location, the *`noDataValue`* is returned.

- `int` **`GetTerrainElevationProfile(`**`double` *`latitude_degrees`*, `double` *`longitude_degrees`*, `double*` *`outputProfile`*, `int` *`sizeOutputProfile`*`)`

Gets a terrain elevation profile in meters from the transmitter's location to the specified location using the terrain elevation data source(s) that were specified using the **`SetPrimaryTerrainElevDataSource`**, **`SetSecondaryTerrainElevDataSource`** and **`SetTertirayTerrainElevDataSource`** methods. The profile's terrain elevation points are equally spaced based on the resolution value provided through **`SetTerrainElevDataSamplingResolution`**, with a minimum of 3 profile points. If no terrain elevation data can be obtained at a specific location, 0 is provided.

`sizeOutputProfile` must specify the maximum number of `double` values that can be safely copied at *`outputProfile`*. When the output profile size is insufficient to fully contain the terrain elevation profile, no data is copied at *`outputProfile`* and the required size (in number of `double` values) is returned. When the output profile size is sufficient, the terrain elevation profile is copied at *`outputProfile`* and the number of written `double` values is returned.

Note: The Python-wrapper version of this method handles the memory allocation internally. It takes latitude_degrees and longitude_degrees as inputs and returns an array.array containing the profile points.

Land cover data methods

- `void SetPrimaryLandCoverDataSource(LandCoverDataSource landCoverSource)`
- `LandCoverDataSource GetPrimaryLandCoverDataSource()`
- `void SetSecondaryLandCoverDataSource(LandCoverDataSource landCoverSource)`
- `LandCoverDataSource GetSecondaryLandCoverDataSource()`

Sets/gets the source(s) for the land cover data. The primary source is the first source CRC-COVLIB tries to get the land cover data from. If no land cover data can be obtained from the primary source for a given location, CRC-COVLIB will try to get it from the secondary source.

It is not always useful to specify land cover data source(s) depending on the propagation model being used. The Longley-Rice, eHata and free space propagation models do not make any use of land cover data. The ITU-R P.1812 propagation model will only make use of land cover data when the surface profile method is set to **P1812_ADD_REPR_CLUTTER_HEIGHT** (see **SetITURP1812SurfaceProfile-Method**). The ITU-R P.452-17 propagation model will only make use of land cover data when the height-gain model mode is set to either **P452_USE_CLUTTER_PROFILE** or **P452_USE_CLUTTER_AT_ENDPOINT** (see **SetITURP452HeightGainModelMode**). And the ITU-R P.452-18 propagation model will only make use of land cover data when the surface profile method is set to **P452_ADD_REPR_CLUTTER_HEIGHT** (see **SetITURP452SurfaceProfileMethod**).

`LandCoverDataSource` is an enumeration type that may take one of the following values:

```
LAND_COVER_NONE           = 200
LAND_COVER_GEOTIFF        = 201
LAND_COVER_ESA_WORLDCOVER = 202
LAND_COVER_CUSTOM         = 203
LAND_COVER_NRCAN          = 204
```

LAND_COVER_NONE
No land cover data source.

LAND_COVER_ESA_WORLDCOVER
Compatible files for this source can be obtained at:
https://viewer.esa-worldcover.org/worldcover/
Note that a user account is required for the download. Only the ESA_WORLDCOVER_10M_MAP product files are required (no need to download the ESA_WORLDCOVER_10M_INPUTQUALITY product files). Files should be unzipped and their location on disk should be specified using the **SetLandCoverDataSourceDirectory** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible *.tif files.

Default mappings exist between ESA WorldCover's classes and some of the propagation models' clutter categories. There are as follows:

ESA WorldCover class	ITU-R P.1812 Clutter Category
10 (Tree cover)	**P1812_URBAN_TREES_FOREST**

50 (Built-up)	**`P1812_URBAN_TREES_FOREST`**
80 (Permanent water bodies)	**`P1812_WATER_SEA`**
All others	**`P1812_OPEN_RURAL`**

ESA WorldCover class	ITU-R P.452-17 Height-Gain Model Clutter Category
10 (Tree cover)	**`P452_HGM_MIXED_TREE_FOREST`**
20 (Shrubland)	**`P452_HGM_IRREGULARLY_SPACED_SPARSE_TREES`**
40 (Cropland)	**`P452_HGM_HIGH_CROP_FIELDS`**
50 (Built-up)	**`P452_HGM_SUBURBAN`**
All others	**`P452_HGM_OTHER`**

ESA WorldCover class	ITU-R P.452-18 Clutter Category
10 (Tree cover)	**`P452_URBAN_TREES_FOREST`**
50 (Built-up)	**`P452_URBAN_TREES_FOREST`**
80 (Permanent water bodies)	**`P452_WATER_SEA`**
All others	**`P452_OPEN_RURAL`**

Default mappings can be modified using the **`SetLandCoverClassMapping`** and **`SetDefaultLandCover-ClassMapping`** methods.

`LAND_COVER_GEOTIFF`
Use provided GeoTIFF files as land cover data source. Location of files on disk should be specified using the **`SetLandCoverDataSourceDirectory`** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible *.tif files. Only a few coordinate reference systems are currently supported, namely “WGS 84” (EPSG:4326), “NAD83” (EPSG:4269), “NAD83(CSRS)” (EPSG:4617), “NAD83(CSRS98)” (EPSG:4140), “WGS 84 / UTM zone [num]”, “NAD83 / UTM zone [num]”, “NAD83(CSRS) / UTM zone [num]” and “NAD83(CSRS) / Canada Atlas Lambert” (EPSG:3979).

`LAND_COVER_CUSTOM`
Uses land cover data submitted trough calls to the **`AddCustomLandCoverData`** method.

`LAND_COVER_NRCAN`
This product consists in a single large *tif file (1.96 GB) that can be obtained at:
https://open.canada.ca/data/en/dataset/ee1580ab-a23d-4f86-a09b-79763677eb47
under the ‘2020 Land Cover of Canada’ link.
The file’s location on disk should be specified using the **`SetLandCoverDataSourceDirectory`** method.

Default mappings exist between NRCAN’s classes and some of the propagation models’ clutter categories. There are as follows:

NRCAN class	ITU-R P.1812 Clutter Category
1 (Temperate or sub-polar needle leaf forest)	`P1812_URBAN_TREES_FOREST`
2 (Sub-polar taiga needle leaf forest)	`P1812_URBAN_TREES_FOREST`
5 (Temperate or sub-polar broad leaf deciduous forest)	`P1812_URBAN_TREES_FOREST`
6 (Temperate or sub-polar deciduous forest)	`P1812_URBAN_TREES_FOREST`
17 (Urban)	`P1812_URBAN_TREES_FOREST`
18 (Water)	`P1812_WATER_SEA`
All others	`P1812_OPEN_RURAL`

NRCAN class	ITU-R P.452-17 Height-Gain Model Clutter
1 (Temperate or sub-polar needle leaf forest)	`P452_HGM_CONIFEROUS_TREES_IRREGULARLY_SPACED`
2 (Sub-polar taiga needle leaf forest)	`P452_HGM_CONIFEROUS_TREES_IRREGULARLY_SPACED`
5 (Temperate or sub-polar broad leaf deciduous forest)	`P452_HGM_DECIDUOUS_TREES_IRREGULARLY_SPACED`
6 (Temperate or sub-polar deciduous forest)	`P452_HGM_DECIDUOUS_TREES_IRREGULARLY_SPACED`
8 (Temperate or sub-polar Shrubland)	`P452_HGM_IRREGULARLY_SPACED_SPARSE_TREES`
15 (Cropland)	`P452_HGM_HIGH_CROP_FIELDS`
17 (Urban)	`P452_HGM_URBAN`
All others	`P452_HGM_OTHER`

NRCAN class	ITU-R P.452-18 Clutter Category
1 (Temperate or sub-polar needle leaf forest)	`P452_URBAN_TREES_FOREST`
2 (Sub-polar taiga needle leaf forest)	`P452_URBAN_TREES_FOREST`
5 (Temperate or sub-polar broad leaf deciduous forest)	`P452_URBAN_TREES_FOREST`
6 (Temperate or sub-polar deciduous forest)	`P452_URBAN_TREES_FOREST`
17 (Urban)	`P452_URBAN_TREES_FOREST`
18 (Water)	`P452_WATER_SEA`
All others	`P1812_OPEN_RURAL`

Default mappings can be modified using the **`SetLandCoverClassMapping`** and **`SetDefaultLandCover-ClassMapping`** methods.

Default value for the primary land cover data source is `LAND_COVER_NONE`.
Default value for the secondary land cover data source is `LAND_COVER_NONE`.

- **void SetLandCoverDataSourceDirectory(LandCoverDataSource** *landCoverSource*,
  const char* *directory*, **bool** *useIndexFile*=**false**, **bool** *overwriteIndexFile*=**false**)
- **const char* GetLandCoverDataSourceDirectory(LandCoverDataSource** *landCoverSource*)

Sets/gets the directory associated with the specified land cover data source.

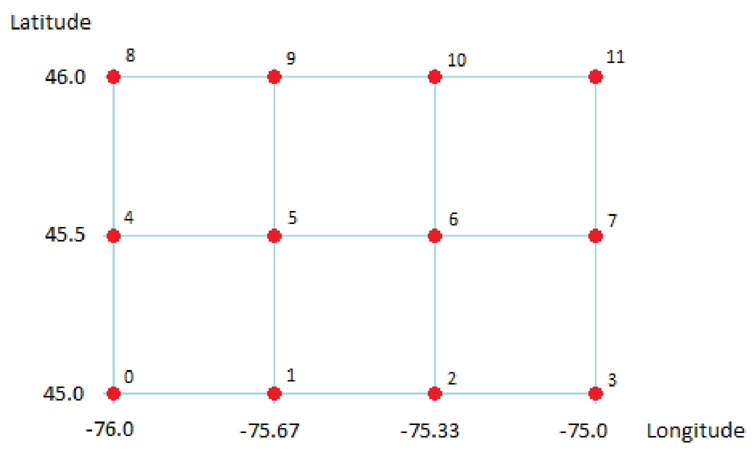
For land cover data sources **LAND_COVER_ESA_WORLDCOVER** and **LAND_COVER_GEOTIFF**, an index file can be used by setting the *useIndexFile* parameter to **true**. This may be useful for directories containing a great number of files in order to prevent CRC-COVLIB from reading the whole directory and sub-directories content each time the **SetLandCoverDataSourceDirectory** method is called. This option creates an index file the first time the directory content is read and subsequently read that single file only as information source. Note that when directories or files are added, removed or modified within a directory using an index file, that index file needs to be regenerated in order for CRC-COVLIB to see the changes. This can be achieved by manually deleting the index file or by setting the *overwriteIndexFile* parameter to **true**. For land cover data sources other than those mentioned above, the *useIndexFile* and *overwriteIndexFile* parameters have no effect.

Default value for the directory of any land cover data source is an empty string.

- **bool AddCustomLandCoverData(**
  double *lowerLeftCornerLat_degrees*, **double** *lowerLeftCornerLon_degrees*,
  double *upperRightCornerLat_degrees*, **double** *upperRightCornerLon_degrees*,
  int *numHorizSamples*, **int** *numVertSamples*, **const short*** *landCoverData*,
  bool *defineNoDataValue*=**false**, **float** *noDataValue*=0)

Adds custom land cover data. *landCoverData* should point to a contiguous array of *numHorizSamples* times *numVertSamples* land cover class values. Land cover class samples are assumed to be equally spaced (i.e. spacing being measured in degrees) along latitude and longitude lines. The first land cover class value pointed to by *landCoverData* is assumed to be geographically located at the specified lower left corner position. The list should then continue first horizontally and then up, up to the specified upper right corner position. For example:

```
AddCustomLandCoverData(45.0, -76.0, 46.0, -75.0, 4, 3, arrayOf12Shorts);
```



If parts of the specified rectangular area do not have valid land cover information, this may be specified by using a “no data” value within *landCoverData*. If a specific value within *landCoverData* must be interpreted as “no data”, this should be indicated by setting the *defineNoDataValue* parameter to **true** and the *noDataValue* parameter to this value. When *defineNoDataValue* is set to **false**, *noDataValue* is ignored.

AddCustomLandCoverData allocates new memory within the **Simulation** object and copies the data pointed to by *landCoverData*. The data pointed to by *landCoverData* is therefore not required to be preserved after the call to **AddCustomLandCoverData**. The copied data is preserved until the **Simulation** object is destroyed (via **Release**) or until a call to **ClearCustomLandCoverData** on the same **Simulation** object.

Returns **false** on failure to allocate sufficient memory, **true** otherwise.

- **void ClearCustomLandCoverData()**

Removes all custom land cover data previously added through **AddCustomLandCoverData** and releases corresponding memory.

- **int GetLandCoverClass(double** *latitude_degrees*, **double** *longitude_degrees*)

Gets the land cover class id at the specified location using the land cover data source(s) that were specified using the **SetPrimaryLandCoverDataSource** and **SetSecondaryLandCoverDataSource** methods. If no land cover data can be obtained at the specified location, -1 is returned.

- **int GetLandCoverClassProfile(double** *latitude_degrees*, **double** *longitude_degrees*, **int*** *outputProfile*, **int** *sizeOutputProfile*)

Gets a land cover class id profile from the transmitter's location to the specified location using the land cover data source(s) that were specified using the **SetPrimaryLandCoverDataSource** and **SetSecondaryLandCoverDataSource** methods. The profile's points are equally spaced based on the resolution value provided through **SetTerrainElevDataSamplingResolution**, with a minimum of 3 profile points. If no land cover class data can be obtained at a specific location, -1 is provided.

sizeOutputProfile must specify the maximum number of `int` values that can be safely copied at *outputProfile*. When the output profile size is insufficient to fully contain the land cover class profile, no data is copied at *outputProfile* and the required size (in number of `int` values) is returned. When the output profile size is sufficient, the land cover class profile is copied at *outputProfile* and the number of written `int` values is returned.

Note: The Python-wrapper version of this method handles the memory allocation internally. It takes latitude_degrees and longitude_degrees as inputs and returns an array.array containing the profile points.

- `void` **SetLandCoverClassMapping**(`LandCoverDataSource` *landCoverSource*, `int` *sourceClass*, `PropagationModel` *propagationModel*, `int` *modelValue*)
- `int` **GetLandCoverClassMapping**(`LandCoverDataSource` *landCoverSource*, `int` *sourceClass*, `PropagationModel` *propagationModel*)

When land cover data is used in a simulation, the land cover class ids need to be mapped to values that are usable by the currently selected propagation model. The **SetLandCoverClassMapping** method is used to specify the mapping between a land cover source's class id and a propagation model's usable value.

For the ITU-R P.1812 and ITU-R P.452-18 propagation models, land cover class ids can either be mapped to clutter categories or to representative clutter heights in meters. For example, to map the "Permanent water bodies" class (id 80) from ESA WorldCover to the "water/sea" clutter category from the ITU-R P.1812 propagation model, use the following:
`SetLandCoverClassMapping(LAND_COVER_ESA_WORLDCOVER, 80, ITU_R_P_1812, P1812_WATER_SEA).`

Alternately, to map the "Permanent water bodies" class (id 80) from ESA WorldCover to a representative clutter height of 0 m, use the following:
`SetLandCoverClassMapping(LAND_COVER_ESA_WORLDCOVER, 80, ITU_R_P_1812, 0).`

The choice whether to interpret the specified *modelValues* as clutter categories or as representative clutter heights is made through the **SetITURP1812LandCoverMappingType** or **SetITURP452LandCover-MappingType** (ITU-R P.452-18) method.

For the ITU-R P.452-17 propagation model, land cover class ids are always mapped to the model's height-gain model clutter categories.

- `void` **SetDefaultLandCoverClassMapping**(`LandCoverDataSource` *landCoverSource*, `PropagationModel` *propagationModel*, `int` *modelValue*)

- **int GetDefaultLandCoverClassMapping**(LandCoverDataSource *landCoverSource*, PropagationModel *propagationModel*)

Sets/gets the default mapping between the specified land cover data source and propagation model (this will be used to map all land cover source's class ids not explicitly mapped with a call to **SetLandCoverClassMapping**).

For example, when using **SetDefaultLandCoverClassMapping**(LAND_COVER_ESA_WORLDCOVER, ITU_R_P_1812, P1812_OPEN_RURAL), any land cover class id from ESA WorldCover that has not been mapped to a specific ITU-R P.1812 clutter category using **SetLandCoverClassMapping** will be mapped to the "open/rural" clutter category. As mentioned previously, the *modelValue* can alternately be interpreted as a representative clutter height in meters, depending on the selection made through **SetITURP1812LandCoverMappingType** or **SetITURP452LandCoverMappingType** (ITU-R P.452-18). For ITU-R P.452-17, *modelValue* should always be interpreted as one of the model's height-gain model clutter categories.

- **void ClearLandCoverClassMappings**(LandCoverDataSource *landCoverSource*, PropagationModel *propagationModel*)

Deletes all land cover class mappings between the specified land cover source and propagation model. This clearing includes any mapping that has been done using both **SetDefaultLandCoveClassMapping** and **SetLandCoverClassMapping**.

- **int GetLandCoverClassMappedValue**(double *latitude_degrees*, double *longitude_degrees*, PropagationModel *propagationModel*)

This method first gets the land cover class id from the source(s) that were specified using the **SetPrimaryLandCoverDataSource** and **SetSecondaryLandCoverDataSource** methods. It then returns the corresponding propagation model's usable value from the mappings specified through **SetLand-CoverClassMapping** and **SetDefaultLandCoverClassMapping**, or through any existing default mapping if none were otherwise specified. If no land cover data could be obtained or no mapping could be applied, a sensible default value is returned based on the propagation model (e.g. 2 [P1812_OPEN_RURAL] for ITU_R_P_1812), or -1 for propagation models that do not use land cover data for their predictions.

- **int GetLandCoverClassMappedValueProfile**(double *latitude_degrees*, double *longitude_degrees*, PropagationModel *propagationModel*, int* *outputProfile*, int *sizeOutputProfile*)

Gets a profile of mapped land cover classes from the transmitter's location to the specified location. A mapped land cover class is a land cover class value originating from a data file that has been converted to a recognizable and usable value by the specified propagation model. The output profile's points are equally spaced based on the resolution value provided through **SetTerrainElevDataSampling-Resolution**, with a minimum of 3 profile points. See **GetLandCoverClassMappedValue**'s description for more details on how the output profile's values are obtained.

sizeOutputProfile must specify the maximum number of **int** values that can be safely copied at *outputProfile*. When the output profile size is insufficient to fully contain the mapped land cover class profile, no data is copied at *outputProfile* and the required size (in number of **int** values) is returned. When the output profile size is sufficient, the mapped land cover class profile is copied at *outputProfile* and the number of written **int** values is returned.

Note: The Python-wrapper version of this method handles the memory allocation internally. It takes latitude_degrees, longitude_degrees and propagatioModel as inputs and returns an array.array containing the profile points.

Surface elevation data methods

- `void SetPrimarySurfaceElevDataSource(SurfaceElevDataSource surfaceElevSource)`
- `SurfaceElevDataSource GetPrimarySurfaceElevDataSource()`
- `void SetSecondarySurfaceElevDataSource(SurfaceElevDataSource surfaceElevSource)`
- `SurfaceElevDataSource GetSecondarySurfaceElevDataSource()`
- `void SetTertiarySurfaceElevDataSource(SurfaceElevDataSource surfaceElevSource)`
- `SurfaceElevDataSource GetTertiarySurfaceElevDataSource()`

Sets/gets the source(s) for the surface elevation data (i.e. ground level + clutter height with no distinction between the two). The primary source is the first source CRC-COVLIB tries to get the surface elevation data from. If surface elevation data cannot be obtained from the primary source for a given location, CRC-COVLIB will try to get it from the secondary source. Similarly, when no surface elevation data can be obtained from both the primary and the secondary sources, CRC-COVLIB will use the tertiary source. Where no surface elevation data can be obtained at all, a surface elevation value of 0 meter is provided to the propagation model. Note that CRC-COVLIB does not make any correction for using different vertical datums.

It is not always useful to specify surface elevation data source(s) depending on the propagation model being used. The Longley-Rice, eHata, free space and ITU-R P.452-17 propagation models do not make any use of surface elevation data. The ITU-R P.1812 propagation model will only make use of surface elevation data when the surface profile method is set to **`P1812_USE_SURFACE_ELEV_DATA`** (see **`SetITURP1812SurfaceProfileMethod`**). The ITU-R P.452-18 propagation model will only make use of surface elevation data when the surface profile method is set to **`P452_EXPERIMENTAL_USE_OF_-SURFACE_ELEV_DATA`** (see **`SetITURP452SurfaceProfileMethod`**).

`SurfaceElevDataSource` is an enumeration type that may take one of the following values:

```
SURF_ELEV_NONE            = 100
SURF_ELEV_SRTM            = 101
SURF_ELEV_CUSTOM          = 102
SURF_ELEV_NRCAN_CDSM      = 103
SURF_ELEV_NRCAN_HRDEM_DSM = 104
SURF_ELEV_GEOTIFF         = 105
SURF_ELEV_NRCAN_MRDEM_DSM = 106
```

`SURF_ELEV_NONE`
No surface elevation data source.

`SURF_ELEV_SRTM` (NASA Shuttle Radar Topography Mission)
Compatible 1 arcsecond resolution files for this source can be obtained at:
https://lpdaac.usgs.gov/products/srtmgl1v003/
https://lpdaac.usgs.gov/products/nasadem_hgtv001/
https://dwtkns.com/srtm30m/
https://step.esa.int/auxdata/dem/SRTMGL1/

Compatible 3 arcseconds resolution files for this source can be obtained at:

https://lpdaac.usgs.gov/products/srtmgl3v003/

Compatible 30 arcseconds resolution files for this source can be obtained at:
https://lpdaac.usgs.gov/products/srtmgl30v021/
https://web.archive.org/web/20170124235811/https://dds.cr.usgs.gov/srtm/version2_1/SRTM30/
http://www.webgis.com/terr_world.html

Files must be unzipped and their location on disk should be specified using the **SetSurfaceElevDataSourceDirectory** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible files (the *.hgt extension is usually used for 1 and 3 arcseconds data and the *.dem extension for 30 arcseconds data). File names specifying start location in latitude and longitude must be preserved. Note that data files originating from the Shuttle Radar Topography Mission (SRTM) may also be made available in the GeoTIFF (*.tif) format. In that case, **SURF_ELEV_GEOTIFF** must be specified as the source instead of **SURF_ELEV_SRTM**.

SURF_ELEV_CUSTOM
Uses surface elevation data submitted trough calls to the **AddCustomSurfaceElevData** method.

SURF_ELEV_NRCAN_CDSM (Natural Resources Canada / Canadian Digital Surface Model)
Compatible files for this source can be obtained at:
https://ftp.maps.canada.ca/pub/nrcan_rncan/elevation/cdsm_mnsc/
https://download-telecharger.services.geo.ca/pub/nrcan_rncan/elevation/cdsm_mnsc/
Files should be unzipped and their location on disk should be specified using the **SetSurfaceElevDataSourceDirectory** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible *.tif files. Other file types (*.pdf, *.xml) may be present but are not required nor used. ***Note: CDSM is now a legacy product, a better option is to use the Medium Resolution Digital Elevation Model (MRDEM) product (see* SURF_ELEV_NRCAN_MRDEM_DSM *below).***

SURF_ELEV_NRCAN_HRDEM_DSM (Natural Resources Canada / High Resolution Digital Elevation Model / Digital Surface Model)
Compatible files for this source can be obtained at:
https://ftp.maps.canada.ca/pub/elevation/dem_mne/highresolution_hauteresolution/dsm_mns/
https://download-telecharger.services.geo.ca/pub/elevation/dem_mne/highresolution_hauteresolution/dsm_mns/
Location of files on disk should be specified using the **SetSurfaceElevDataSourceDirectory** method. CRC-COVLIB will search the specified directory and any sub-directories for compatible *.tif files. Note that the files that may be used are those with names starting with 'dsm_' (example: dsm_1m_utm18_w_1_102.tif).
The following tool may also be used to search for and download HRDEM files:
https://search.open.canada.ca/openmap/957782bf-847c-4644-a757-e383c0057995
Note that the ArticDEM mosaic files (in the polar stereographic North coordinate system) are not supported.
For additional details on this product, please see:
https://open.canada.ca/data/en/dataset/957782bf-847c-4644-a757-e383c0057995
https://open.canada.ca/data/en/dataset/0fe65119-e96e-4a57-8bfe-9d9245fba06b

SURF_ELEV_GEOTIFF
Use provided GeoTIFF files as surface elevation source. Location of files on disk should be specified using the **SetSurfaceElevDataSourceDirectory** method. CRC-COVLIB will search the specified directory and

any sub-directories for compatible *.tif files. Only a few coordinate reference systems are currently supported, namely "WGS 84" (EPSG:4326), "NAD83" (EPSG:4269), "NAD83(CSRS)" (EPSG:4617), "NAD83(CSRS98)" (EPSG:4140), "WGS 84 / UTM zone [num]", "NAD83 / UTM zone [num]", "NAD83(CSRS) / UTM zone [num]" and "NAD83(CSRS) / Canada Atlas Lambert" (EPSG:3979).

SURF_ELEV_NRCAN_MRDEM_DSM (Natural Resources Canada / Medium Resolution Digital Elevation Model / Digital Surface Model)
This product consists in a large cloud optimized GeoTIFF file (55.6 GB) available at:
https://datacube-prod-data-public.s3.ca-central-1.amazonaws.com/store/elevation/mrdem/mrdem-30/mrdem-30-dsm.tif
CRC-COVLIB can either use the whole file as is or extracts from it. In both cases, the file(s)' location on disk should be specified using the **SetSurfaceElevDataSourceDirectory** method. For additional details on this product, please see:
https://open.canada.ca/data/en/dataset/18752265-bda3-498c-a4ba-9dfe68cb98da

Note: CRC-COVLIB's python package contains functions to facilitate the download of extracts from the CDEM, HRDEM and MRDEM products. They are available under the crc_covlib.helper.datacube_canada module.

Default value for the primary surface elevation data source is **SURF_ELEV_NONE**.
Default value for the secondary surface elevation data source is **SURF_ELEV_NONE**.
Default value for the tertiary surface elevation data source is **SURF_ELEV_NONE**.

- `void` **SetSurfaceElevDataSourceDirectory**(`SurfaceElevDataSource` *surfaceElevSource*, `const char*` *directory*, `bool` *useIndexFile*=`false`, `bool` *overwriteIndexFile*=`false`)
- `const char*` **GetSurfaceElevDataSourceDirectory**(`SurfaceElevDataSource` *surfaceElevSource*)

Sets/gets the directory associated with the specified surface elevation data source.

For surface elevation data sources **SURF_ELEV_NRCAN_CDSM**, **SURF_ELEV_NRCAN_HRDEM_DSM** and **SURF_ELEV_GEOTIFF**, an index file can be used by setting the *useIndexFile* parameter to **true**. This may be useful for directories containing a great number of files in order to prevent CRC-COVLIB from reading the whole directory and sub-directories content each time the **SetSurfaceElevDataSource-Directory** method is called. This option creates an index file the first time the directory content is read and subsequently read that single file only as information source. Note that when directories or files are added, removed or modified within a directory using an index file, that index file needs to be regenerated in order for CRC-COVLIB to see the changes. This can be achieved by manually deleting the index file or by setting the *overwriteIndexFile* parameter to **true**. For surface elevation data sources other than those mentioned above, the *useIndexFile* and *overwriteIndexFile* parameters have no effect.

Default value for the directory of any surface elevation data source is an empty string.

- `void` **SetSurfaceElevDataSourceSamplingMethod**(`SurfaceElevDataSource` *surfaceElevSource*, `SamplingMethod` *samplingMethod*)

- `SamplingMethod GetSurfaceElevDataSourceSamplingMethod( SurfaceElevDataSource surfaceElevSource)`

Sets/gets the sampling method for the specified surface elevation data source. `SamplingMethod` is an enumeration type that may take one of the following values:

```
NEAREST_NEIGHBOR       = 0
BILINEAR_INTERPOLATION = 1
```

The sampling method applies when reading surface elevation data files. The nearest neighbor method requires reading one value from file while the bilinear interpolation method requires four. For this reason the nearest neighbor method is normally faster. Default value is `BILINEAR_INTERPOLATION` for all surface elevation data sources.

- `void SetSurfaceAndTerrainDataSourcePairing(bool usePairing)`
- `bool GetSurfaceAndTerrainDataSourcePairing()`

Sets/gets the state of pairing utilization between surface and terrain elevation data sources.

When *usePairing* is set to **true**, the primary surface elevation data source is paired with the primary terrain elevation data source, the secondary surface source with the secondary terrain source, and the tertiary surface source with the tertiary terrain source, forming up to three pairs. For any surface elevation data to be used at a given location, terrain elevation data at the same location for the corresponding paired source must also be available, and vice-versa. Otherwise the data is considered to be missing for both paired sources.

Let's say we have the following configuration and that pairing is set to **true**:

	Surface elevation data source	Terrain elevation data source
Primary	`SURF_ELEV_NRCAN_HRDEM_DSM`	`TERR_ELEV_NRCAN_HRDEM_DTM`
Secondary	`SURF_ELEV_NRCAN_CDSM`	`TERR_ELEV_NRCAN_CDEM`
Tertiary	`SURF_ELEV_NONE`	`TERR_ELEV_NONE`

If we try to get the surface elevation at location (lat=45, lon-75) for example, CRC-COVLIB will first try to get it from the HRDEM DSM product (as it is the primary source). It will also try to get the terrain elevation at location (lat=45, lon-75) using the HRDEM DTM product. If either or both the surface elevation and terrain elevation could not be obtained at that location, CRC-COVLIB will continue with using the secondary and tertiary sources in a same manner, until both surface and terrain elevation could be obtained from a same pair. If even that is not possible, either 0 meter or the specified "no data" value will be returned for the surface elevation depending on the context.

This pairing option ensures that surface and terrain elevation data products that are intended to be used together for optimal accuracy will never be mixed with other products. This may be useful when calculating results with a propagation model that uses both surface and elevation data for example.

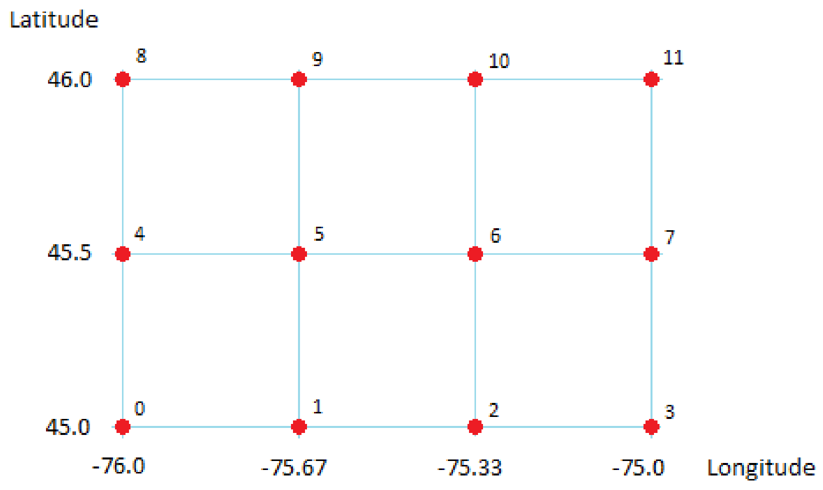
However in practice the use of the pairing option is not necessary when products that are meant to be used together already cover the exact same area (if your HRDEM DSM files cover the exact same region covered by your HRDEM DTM files for example).

Default value for pairing utilization is `false`.

-
  ```
  bool AddCustomSurfaceElevData(
    double lowerLeftCornerLat_degrees, double lowerLeftCornerLon_degrees,
    double upperRightCornerLat_degrees, double upperRightCornerLon_degrees,
    int numHorizSamples, int numVertSamples, const float* surfaceElevData_meters,
    bool defineNoDataValue=false, float noDataValue=0)
  ```

Adds custom surface elevation data. *surfaceElevData_meters* should point to a contiguous list of *numHorizSamples* times *numVertSamples* surface elevation values. Surface elevation samples are assumed to be equally spaced (i.e. spacing being measured in degrees) along latitude and longitude lines. The first surface elevation value pointed to by *surfaceElevData_meters* is assumed to be geographically located at the specified lower left corner position. The list should then continue first horizontally and then up, up to the specified upper right corner position. For example:

```
AddCustomSurfaceElevData(45.0, -76.0, 46.0, -75.0, 4, 3, arrayOf12Floats);
```



If parts of the specified rectangular area do not have valid surface elevation information, this may be specified by using a "no data" value within *surfaceElevData_meters*. If a specific value within *surfaceElevData_meters* must be interpreted as "no data", this should be indicated by setting the *defineNoDataValue* parameter to `true` and the *noDataValue* parameter to this value. When *defineNoDataValue* is set to `false`, *noDataValue* is ignored.

AddCustomSurfaceElevData allocates new memory within the `Simulation` object and copies the data pointed to by *surfaceElevData_meters*. The data pointed to by *surfaceElevData_meters* is therefore not required to be preserved after the call to **AddCustomSurfaceElevData**. The copied data is preserved until the `Simulation` object is destroyed (via **`Release`**) or until a call to **`ClearCustom-SurfaceElevData`** on the same `Simulation` object.

Returns `false` on failure to allocate sufficient memory, `true` otherwise.

- `void` **`ClearCustomSurfaceElevData`**`()`

Removes all custom surface elevation data previously added through **`AddCustomTerrainElevData`** and releases corresponding memory.

- `double` **`GetSurfaceElevation`**`(double` *`latitude_degrees`*`, double` *`longitude_degrees`*`,`
  `double` *`noDataValue`*`=0)`

Gets the terrain surface elevation in meters at the specified location using the surface elevation data source(s) that were specified using the **`SetPrimarySurfaceElevDataSource`**, **`SetSecondarySurface-ElevDataSource`** and **`SetTertiraySurfaceElevDataSource`** methods. If no surface elevation data can be obtained at the specified location, the *`noDataValue`* is returned.

- `int` **`GetSurfaceElevationProfile`**`(double` *`latitude_degrees`*`, double` *`longitude_degrees`*`,`
  `double*` *`outputProfile`*`, int` *`sizeOutputProfile`*`)`

Gets a surface elevation profile in meters from the transmitter's location to the specified location using the surface elevation data source(s) that were specified using the **`SetPrimarySurfaceElevDataSource`**, **`SetSecondarySurfaceElevDataSource`** and **`SetTertiraysurfaceElevDataSource`** methods. The profile's surface elevation points are equally spaced based on the resolution value provided through **`SetTerrainElevDataSamplingResolution`**, with a minimum of 3 profile points. If no surface elevation data can be obtained at a specific location, 0 is provided.

`sizeOutputProfile` must specify the maximum number of `double` values that can be safely copied at *`outputProfile`*. When the output profile size is insufficient to fully contain the surface elevation profile, no data is copied at *`outputProfile`* and the required size (in number of `double` values) is returned. When the output profile size is sufficient, the surface elevation profile is copied at *`outputProfile`* and the number of written `double` values is returned.

Note: The Python-wrapper version of this method handles the memory allocation internally. It takes latitude_degrees and longitude_degrees as inputs and returns an array.array containing the profile points.

Reception area methods

- `void SetReceptionAreaCorners(double lowerLeftCornerLat_degrees, double lowerLeftCornerLon_degrees, double upperRightCornerLat_degrees, double upperRightCornerLon_degrees)`
- `double GetReceptionAreaLowerLeftCornerLatitude()`
- `double GetReceptionAreaLowerLeftCornerLongitude()`
- `double GetReceptionAreaUpperRightCornerLatitude()`
- `double GetReceptionAreaUpperRightCornerLongitude()`

Sets/gets the reception area location. The reception area is a rectangular grid of reception points at which simulation results (field strength, path loss, transmission loss or received power) are calculated when calling the **GenerateReceptionAreaResults** method.

- `void SetReceptionAreaNumHorizontalPoints(int numPoints)`
- `int GetReceptionAreaNumHorizontalPoints()`
- `void SetReceptionAreaNumVerticalPoints(int numPoints)`
- `int GetReceptionAreaNumVerticalPoints()`

Sets/gets the number of reception points on the horizontal/vertical side of the reception area. The total number of reception points in the reception area is given by the number of horizontal points multiplied by the number of vertical points. When either the horizontal or vertical number of points is modified, any data the reception area may contain will be cleared to all zeros.

numPoints must be equal or greater than 2. Default value for both the number of horizontal and vertical points is 60 (for a total of 3600 reception points).

Result type selection methods

- `void SetResultType(ResultType resultType)`
- `ResultType GetResultType()`

Sets/gets the simulation's result type. `ResultType` is an enumeration type that may take one of the following values:

```
FIELD_STRENGTH_DBUVM = 1
PATH_LOSS_DB         = 2
TRANSMISSION_LOSS_DB = 3
RECEIVED_POWER_DBM   = 4
```

Simulation results are generated by calling one of the **GenerateReceptionPointResult**, **GenerateReceptionPointDetailedResult**, **GenerateReceptionAreaResults**, **GenerateProfile-ReceptionPointResult** and **ExportProfilesToCsvFile** methods. Path loss (dB) is the value computed by propagation models and possibly complemented by other loss models (clutter, buildings, etc.). Other results are calculated as follows:

- field strength (dBµV/m) = Ptx - Lp + Gtx - Ltx + 137.21 + $20\log_{10}(f)$
- transmission loss (dB) = Lp - Gtx + Ltx - Grx + Lrx
- received power (dBm) = Ptx - Lp + Gtx - Ltx + Grx – Lrx

```
where
  Ptx = transmit power output (dBm)
  Lp  = propagation loss or path loss (dB)
  Gtx = transmitter antenna gain (dBi)
  Grx = receiver antenna gain (dBi)
  Ltx = transmitter losses (dB)
  Lrx = receiver losses (dB)
  f   = frequency (GHz)
```

Default value for the result type is `FIELD_STRENGTH_DBUVM`.

Coverage display fills methods

- `void` **`AddCoverageDisplayFill`**`(double` *`fromValue`*`, double` *`toValue`*`, int` *`rgbColor`*`)`

Adds a new coverage display fill. After results have been generated for a defined reception area, a visual representation of the results can be exported to a vector file using the **`ExportReceptionArea-ResultsToMifFile`** or **`ExportReceptionAreaResultsToKmlFile`** methods. Areas where results fall between *`fromValue`* and *`toValue`* will be colored using the specified *`rgbColor`*. Ranges of values from one display fill to another are allowed to overlap. By default, the simulation contains three coverage display fills: (45, 60, 0x5555FF), (60, 75, 0x0000FF) and (75, 100, 0x000088).

- `void` **`ClearCoverageDisplayFills`**`()`

Deletes all coverage display fills.

- `int` **`GetCoverageDisplayNumFills`**`()`

Gets the number of currently defined coverage display fills.

- `double` **`GetCoverageDisplayFillFromValue`**`(int` *`index`*`)`
- `double` **`GetCoverageDisplayFillToValue`**`(int` *`index`*`)`
- `int` **`GetCoverageDisplayFillColor`**`(int` *`index`*`)`

Gets details of the coverage display fill at the specified zero-based index. *`index`* must be between 0 and **`GetCoverageDisplayNumFills`**`()-1`.

Computing and accessing results methods

- `double GenerateReceptionPointResult(double latitude_degrees,`
  `double longitude_degrees)`

Returns the computed result (field strength, path loss, transmission loss or received power) at the specified location based on current simulation parameters.

- `ReceptionPointDetailedResult GenerateReceptionPointDetailedResult(`
  `double latitude_degrees, double longitude_degrees)`

Returns a structure containing the computed result (field strength, path loss, transmission loss or received power) at the specified location based on current simulation parameters, along with other computed values.

The `ReceptionPointDetailedResult` structure contains the following members:

```
double result;
double pathLoss_dB;
double pathLength_km;
double transmitterHeightAMSL_m;
double receiverHeightAMSL_m;
double transmitterAntennaGain_dBi;
double receiverAntennaGain_dBi;
double azimuthFromTransmitter_degrees;
double azimuthFromReceiver_degrees;
double elevAngleFromTransmitter_degrees;
double elevAngleFromReceiver_degrees;
```

The **pathLength_km** value is the great-circle distance between the transmitter's and receiver's location. The height values for **transmitterHeightAMSL_m** and **receiverHeightAMSL_m** are in meters above mean sea level. They are composed of the antenna height above ground level and the terrain elevation value obtained from the terrain elevation data source(s), when specified.

Contrarily to **GenerateReceptionPointResult**, **GenerateReceptionPointDetailedResult** always computes antenna gains even though they may not be required for computing the selected result type. It also provides the computed azimuths and elevation angles utilized for getting antenna pattern gains. Azimuths are referenced from true north. Elevation angles are referenced from the local horizon, where -90 is the zenith, 0 is the astronomical horizon and +90 is the nadir. They are computed for median refractivity conditions using the antenna heights and the terrain elevation profile between the terminals (if one or more terrain elevation data source(s) were specified for the simulation). In line-of-sight situations, a computed elevation angle will be towards the other terminal's antenna, otherwise it will be towards the radio horizon (i.e. the terrain clearance angle).

- `double GenerateProfileReceptionPointResult(`
  `double latitude_degrees, double longitude_degrees, int numSamples,`
  `const double* terrainElevProfile,`

```
    const int* landCoverClassMappedValueProfile=NULL,
    const double* surfaceElevProfile=NULL,
    const ITURadioClimaticZone* ituRadioClimaticZoneProfile=NULL)
```

Returns the computed result (field strength, path loss, transmission loss or received power) at the specified location based on current simulation parameters. Instead of using the terrain elevation data source(s) specified using the **SetPrimaryTerrainElevDataSource**, **SetSecondaryTerrainElevDataSource** and **SetTertiaryTerrainElevDataSource** methods, this method uses the terrain elevation profile specified as parameter.

Similarly, a land cover mapped value profile may be specified. The values contained in this profile must be the mapped values that are directly understood by the selected propagation model. For the ITU-R P.1812 and ITRU-R P.452-18 propagation models, this can either be clutter categories (i.e. values defined under the **P1812ClutterCategory** or **P452ClutterCategory** types) or representative clutter heights in meters, depending on the selection made through **SetITURP1812LandCoverMappingType** or **SetITURP452LandCoverMappingType**. For ITU-R P.452-17, this can only be height-gain model clutter categories (i.e. values defined under the **P452HeightGainModelClutterCategory** type).

A surface elevation profile may also be specified. This profile may be used by the ITU-R P.1812 and ITU-R P.452-18 models depending on the selected surface profile method (see **SetITURP1812SurfaceProfileMethod** and **SetITURP452SurfaceProfileMethod**), as well as by the CRC-MLPL and CRC Path Obscura models.

Finally, an ITU radio climatic zones profile may be specified. This profile is utilized by the ITU-R P.1812 and ITU-R P.452 (both versions) propagation models. **ITURadioClimaticZone** is an enumeration type that may take one of the following values:

```
ITU_COASTAL_LAND = 3
ITU_INLAND       = 4
ITU_SEA          = 1
```

When a profile is set to NULL (including *terrainElevProfile* which may be set to NULL just like the others), its samples are taken from the usual primary/secondary/tertiary source(s) if required by the selected propagation model. For the ITU radio climatic zones profile, its samples are obtained using the specified file from **SetITURP1812RadioClimaticZonesFile** or **SetITURP452RadioClimaticZonesFile**, if required.

latitude_degrees and *longitude_degrees* indicate the reception point location. *numSamples* indicates the number of samples in *terrainElevProfile*, *landCoverClassMappedValueProfile*, *surfaceElevProfile* and *ituRadioClimaticZoneProfile* when not NULL (i.e. if more than one profile is specified, they must all contain the same number of samples). The terrain and surface elevation samples must be in meters.

-
  ```
  ReceptionPointDetailedResult GenerateProfileReceptionPointDetailedResult(
    double latitude_degrees, double longitude_degrees, int numSamples,
    const double* terrainElevProfile,
    const int* landCoverClassMappedValueProfile=NULL,
    const double* surfaceElevProfile=NULL,
    const ITURadioClimaticZone* ituRadioClimaticZoneProfile=NULL)
  ```

Returns a structure containing the computed result (field strength, path loss, transmission loss or received power) at the specified location based on current simulation parameters, along with other computed values.

The `ReceptionPointDetailedResult` structure contains the following members:

```
double result;
double pathLoss_dB;
double pathLength_km;
double transmitterHeightAMSL_m;
double receiverHeightAMSL_m;
double transmitterAntennaGain_dBi;
double receiverAntennaGain_dBi;
double azimuthFromTransmitter_degrees;
double azimuthFromReceiver_degrees;
double elevAngleFromTransmitter_degrees;
double elevAngleFromReceiver_degrees;
```

For more details on `ReceptionPointDetailedResult`, see **GenerateReceptionPointDetailed-Result**'s description.

When a profile parameter is set to NULL, its samples are taken from the usual primary/secondary/tertiary source(s) if required by the selected propagation model. Otherwise the provided profile is used when generating results. For more details on the profile parameters, see **GenerateProfileReceptionPointResult**'s description.

latitude_degrees and *longitude_degrees* indicate the reception point location. *numSamples* indicates the number of samples in *terrainElevProfile*, *landCoverClassMappedValueProfile*, *surfaceElevProfile* and *ituRadioClimaticZoneProfile* when not NULL (i.e. if more than one profile is specified, they must all contain the same number of samples). The terrain and surface elevation samples must be in meters.

- `void` **GenerateReceptionAreaResults**`()`

Computes results (field strength, path loss, transmission loss or received power) inside the currently defined reception area for current simulation parameters. Execution time for **GenerateReception-AreaResults** may greatly vary depending on simulation parameters like the terrain elevation data sampling resolution and the number of reception points inside the reception area. After calling this method, results generated by a previous call to **GenerateReceptionAreaResults** on the same simulation object will not be accessible anymore.

- `int` **GetGenerateStatus()**

The **GetGenerateStatus** method returns a status value associated with the last call to one of the results generating method (**GenerateReceptionPointResult**, **GenerateReceptionPointDetailedResult**, **GenerateProfileReceptionPointResult**, **GenerateProfileReceptionPointDetailedResult**,

GenerateReceptionAreaResults and **ExportProfilesToCsvFile**). This value may be checked against different flag values of type `GenerateStatus` to determine which one(s) apply. `GenerateStatus` is an enumeration type that defines the following flag values:

```
STATUS_OK                              =   0
STATUS_NO_TERRAIN_ELEV_DATA            =   1
STATUS_SOME_TERRAIN_ELEV_DATA_MISSING  =   2
STATUS_NO_LAND_COVER_DATA              =   4
STATUS_SOME_LAND_COVER_DATA_MISSING    =   8
STATUS_NO_ITU_RCZ_DATA                 =  16
STATUS_SOME_ITU_RCZ_DATA_MISSING       =  32
STATUS_NO_SURFACE_ELEV_DATA            =  64
STATUS_SOME_SURFACE_ELEV_DATA_MISSING  = 128
```

When **GetGenerateStatus** returns 0 (**STATUS_OK**) then no other status value applies. Otherwise the bitwise AND operator (&) can be used on the returned value to check which status value(s) apply. For example:

```
int status = sim->GetGenerateStatus();
if( (status & STATUS_SOME_TERRAIN_ELEV_DATA_MISSING) != 0 )
  cout << "Some terrain elevation data could not be obtained!" << endl;
```

Flag values description:

STATUS_SOME_TERRAIN_ELEV_DATA_MISSING
When this flag is raised, it indicates that at least one terrain elevation value could not be obtained from any of the terrain elevation data sources (primary, secondary and tertiary).

STATUS_NO_TERRAIN_ELEV_DATA
When this flag is raised, it indicates that absolutely no terrain elevation data could be obtained from any of the terrain elevation data sources (primary, secondary and tertiary). This may indicate that a terrain elevation data source directory has not been set or has been wrongly set. It may also be that the reception area does not overlap any of the terrain elevation data files' coverage.

STATUS_SOME_LAND_COVER_DATA_MISSING
When this flag is raised, it indicates that at least one land cover value could not be obtained from any of the land cover data sources (primary, secondary).

STATUS_NO_LAND_COVER_DATA
When this flag is raised, it indicates that absolutely no land cover data could be obtained from any of the land cover data sources (primary, secondary). This may indicate that a land cover data source directory has not been set or has been wrongly set. It may also be that the reception area does not overlap any of the land cover data files' coverage. Another possibility is that no mapping exists between the land cover data source(s) and the selected propagation model. However, the land cover data flags are never raised when the selected propagation model does not make any use of land cover data.

STATUS_SOME_SURFACE_ELEV_DATA_MISSING
When this flag is raised, it indicates that at least one surface elevation value could not be obtained from any of the surface elevation data sources (primary, secondary and tertiary).

STATUS_NO_SURFACE_ELEV_DATA
When this flag is raised, it indicates that absolutely no surface elevation data could be obtained from any of the surface elevation data sources (primary, secondary and tertiary). This may indicate that a

surface elevation data source directory has not been set or has been wrongly set. It may also be that the reception area does not overlap any of the surface elevation data files' coverage. The surface elevation data flags are never raised when the selected propagation model does not make any use of surface elevation data.

STATUS_SOME_ITU_RCZ_DATA_MISSING
When this flag is raised, it indicates that at least one ITU radio climatic zone value could not be obtained from the source file specified using **SetITURP1812RadioClimaticZonesFile** or **SetITURP452Radio-ClimaticZonesFile**.

STATUS_NO_ITU_RCZ_DATA
When this flag is raised, it indicates that absolutely no ITU radio climatic zone value could be obtained from the source file specified using **SetITURP1812RadioClimaticZonesFile** or **SetITURP1812-RadioClimaticZonesFile**. The radio climatic zones data flags are never raised when the selected propagation model does not make any use of radio climatic zones.

Finally, note that flags in general won't be raised if all corresponding data sources are set to NONE, since CRC-COVLIB has been instructed then not to look for any data.

- `double` **`GetReceptionAreaResultValue`**`(int` *`xIndex`*`, int` *`yIndex`*`)`

Once the **GenerateReceptionAreaResults** method has been called, the **GetReceptionArea-ResultValue** can be called to obtain the calculated result at the reception point of specified indexes within the reception area. *xIndex* must be an integer ranging from 0 to the number of horizontal points in the reception area minus 1. *yIndex* must be an integer ranging from 0 to the number of vertical points in the reception area minus 1. Returns 0 when the indexes are out of their valid range.

- `void` **`SetReceptionAreaResultValue`**`(int` *`xIndex`*`, int` *`yIndex`*`, double` *`value`*`)`

Sets or modifies the result at the reception point of specified indexes within the reception area. *xIndex* must be an integer ranging from 0 to the number of horizontal points in the reception area minus 1. *yIndex* must be an integer ranging from 0 to the number of vertical points in the reception area minus 1.

- `double` **`GetReceptionAreaResultValueAtLatLon`**`(double` *`latitude_degrees`*`,`
  `double` *`longitude_degrees`*`)`

Once the **GenerateReceptionAreaResults** method has been called, this method can be called to obtain the calculated result at the specified geographic location within the reception area. The result is obtained by interpolating between the already calculated results of the closest reception points in the reception area. Returns 0 when the specified location is outside of the reception area.

- `double` **`GetReceptionAreaResultLatitude`**`(int` *`xIndex`*`, int` *`yIndex`*`)`
- `double` **`GetReceptionAreaResultLongitude`**`(int` *`xIndex`*`, int` *`yIndex`*`)`

Gets the latitude/longitude coordinate (in degrees) of the reception area's reception point of specified indexes. `xIndex` must be an integer ranging from 0 to the number of horizontal points in the reception area minus 1. `yIndex` must be an integer ranging from 0 to the number of vertical points in the reception area minus 1. Returns 0 when the indexes are out of their valid range.

- `bool` **`ExportReceptionAreaResultsToTextFile`**`(const char* pathname, const char* resultsColumnName=NULL)`

Once the **`GenerateReceptionAreaResults`** method has been called, this method can be called to write the reception area results into a text file. The file will contain a list of all reception points with their location (latitude and longitude) and associated result (field strength, path loss, transmission loss or received power). `pathname` should be a string containing the name of the file to be created or overwritten and may optionally include a full or relative path. `resultsColumnName` may be used to specify the results' column name to be written in the file's header row. If not specified, a default column name will be used. Returns `true` when the file could be created or overwritten, `false` otherwise.

- `bool` **`ExportReceptionAreaResultsToMifFile`**`(const char* pathname, const char* resultsUnits=NULL)`

Once the **`GenerateReceptionAreaResults`** method has been called, this method can be called to write the reception area results into a mif (MapInfo Interchange File) vector file along with its associated mid file for storing attributes. The mif file will contain a graphical representation of the results based on the defined coverage display fills. `pathname` should be a string containing the name of the mif file to be created or overwritten and may optionally include a full or relative path. `resultsUnits` may be used to modify the default unit of measure for results (may be useful when the reception area's content was modified using **`SetReceptionAreaResultValue`** to values of a different unit of measure than what would be expected from the selected result type). Returns `true` when both files could be created or overwritten, `false` otherwise. Mif files can be displayed using most geographic information system (GIS) software.

- `bool` **`ExportReceptionAreaResultsToKmlFile`**`(const char* pathname, double fillOpacity_percent=50, double lineOpacity_percent=50, const char* resultsUnits=NULL)`

Once the **`GenerateReceptionAreaResults`** method has been called, this method can be called to write the reception area results into a kml (Keyhole Markup Language) vector file. The kml file will contain a graphical representation of the results based on the defined coverage display fills. `pathname` should be a string containing the name of the kml file to be created or overwritten and may optionally include a full or relative path. The `fillOpacity_percent` and `lineOpacity_percent` parameters may be used to modify the default opacity of polygon fills and outlines, with values from 0 (fully transparent) to 100 (fully opaque). `resultsUnits` may be used to modify the default unit of measure for results (may be useful when the reception area's content was modified using **`SetReceptionAreaResultValue`** to values of a different unit of measure than what would be expected from the selected result type).Returns `true` when the file could be created or overwritten, `false` otherwise. Kml files can be opened using Google Earth and most geographic information system (GIS) software.

- `bool` **`ExportReceptionAreaResultsToBilFile`**`(const char*` *`pathname`*`)`

Once the **`GenerateReceptionAreaResults`** method has been called, this method can be called to write the reception area results into a bil (Band Interleaved by Line) raster file along with its associated hdr (header) and prj (projection) files. The bil file will contain result values associated with each reception point of the reception area. *`pathname`* should be a string containing the name of the bil file to be created or overwritten and may optionally include a full or relative path. Returns `true` when all three files could be created or overwritten successfully, `false` otherwise. Bil files can be displayed using most geographic information system (GIS) software.

- `bool` **`ExportReceptionAreaTerrainElevationToBilFile`**`(const char*` *`pathname`*`,`
  `int` *`numHorizontalPoints`*`, int` *`numVerticalPoints`*`, bool` *`setNoDataToZero`*`=false)`

Exports terrain elevation data over the area covered by the reception area (specified using **`SetReceptionAreaCorners`**) into a bil (Band Interleaved by Line) raster file along with its associated hdr (header) and prj (projection) files. The method exports *`numHorizontalPoints`* times *`numVerticalPoints`* points using terrain elevation data source(s) previously specified using the **`SetPrimaryTerrainElevDataSource`**, **`SetSecondaryTerrainElevDataSource`** and **`SetTertiary-TerrainElevDataSource`** methods. When *`setNoDataToZero`* is set to `true`, a value of 0 meter is used when no terrain elevation data can be retrieved from the terrain elevation data source(s). Otherwise a “no data” value is used. Returns `true` when all three files could be created or overwritten successfully, `false` otherwise. Bil files can be displayed using most geographic information system (GIS) software.

- `bool` **`ExportReceptionAreaLandCoverClassesToBilFile`**`(const char*` *`pathname`*`,`
  `int` *`numHorizontalPoints`*`, int` *`numVerticalPoints`*`, bool` *`mapValues`*`)`

Exports land cover class data over the area covered by the reception area (specified using **`SetReceptionAreaCorners`**) into a bil (Band Interleaved by Line) raster file along with its associated hdr (header) and prj (projection) files. The method exports *`numHorizontalPoints`* times *`numVerticalPoints`* points using land cover data source(s) previously specified using the **`SetPrimaryLandCoverDataSource`** and **`SetSecondaryLandCoverDataSource`** methods. When *`mapValues`* is set to `true`, any defined mapping between the land cover classes and the currently selected propagation model (via **`SetPropagationModel`**) is applied before the export takes place. See **`SetLandCoverClassMapping`** and **`SetDefaultLandCoverClassMapping`** for more details on land cover class mapping. Returns `true` when all three files could be created or overwritten successfully, `false` otherwise. Bil files can be displayed using most geographic information system (GIS) software.

- `bool` **`ExportReceptionAreaSurfaceElevationToBilFile`**`(const char*` *`pathname`*`,`
  `int` *`numHorizontalPoints`*`, int` *`numVerticalPoints`*`, bool` *`setNoDataToZero`*`=false)`

Exports surface elevation data over the area covered by the reception area (specified using **SetReceptionAreaCorners**) into a bil (Band Interleaved by Line) raster file along with its associated hdr (header) and prj (projection) files. The method exports *numHorizontalPoints* times *numVerticalPoints* points using surface elevation data source(s) previously specified using the **SetPrimarySurfaceElevDataSource**, **SetSecondarySurfaceElevDataSource** and **SetTertiary-SurfaceElevDataSource** methods. When *setNoDataToZero* is set to **true**, a value of 0 meter is used when no surface elevation data can be retrieved from the terrain elevation data source(s). Otherwise a "no data" value is used. Returns **true** when all three files could be created or overwritten successfully, **false** otherwise. Bil files can be displayed using most geographic information system (GIS) software.

- **bool ExportProfilesToCsvFile(const char*** *pathname*, **double** *latitude_degrees*, **double** *longitude_degrees*)

Computes results (field strength, path loss, transmission loss or received power) using current simulation parameters along the path between the transmitter location and the specified reception point at *latitude_degrees* and *longitude_degrees*. This result profile along with other applicable profiles (i.e. terrain elevation, land cover, etc.) are then exported into a csv (Comma-Separated Values) file. *pathname* should be a string containing the name of the csv file to be created or overwritten and may optionally include a full or relative path. Returns **true** when the file could be created or overwritten, **false** otherwise.

The number of points in each profiles is determined by the sampling resolution specified using the **SetTerrainElevDataSamplingResolution** method.

Concurrency

It should be safe to call methods of different instances of the Simulation class in parallel. However it must not be assumed that methods of a same instance may be safely called in parallel.

References

[1] J. Ethier, M. Châteauvert. "Machine Learning-Based Path Loss Modeling With Simplified Features," in IEEE Antennas and Wireless Propagation Letters, vol. 23, no. 11, pp. 3997–4001, 2024. Preprint on arXiv: https://arxiv.org/abs/2405.10006

[2] J. Ethier, M. Châteauvert, R. G. Dempsey, A. Bose. "Path Loss Prediction Using Machine Learning with Extended Features," January 2025. Preprint on arXiv: https://arxiv.org/pdf/2501.08306

[3] Thiele, Lars & Wirth, T & Börner, Kai & Olbrich, Michael & Jungnickel, Volker & Rumold, Juergen & Fritze, Stefan. (2009). Modeling of 3D field patterns of downtilted antennas and their impact on cellular systems.

[4] T. G. Vasiliadis, A. G. Dimitriou and G. D. Sergiadis, "A novel technique for the approximation of 3-D antenna radiation patterns," in IEEE Transactions on Antennas and Propagation, vol. 53, no. 7, pp. 2212-2219, July 2005, doi: 10.1109/TAP.2005.850752.

Contact information

For any question regarding this document or CRC-COVLIB, you may contact the main author at:

martin-pierre.lussier@ised-isde.gc.ca

Personal thanks

Thanks to Jérôme Blais-Morin for his contribution to CRC-COVLIB's original version and to Doris Camiré for her support to this project. Thanks to Bernard Doray for many insightful comments and suggestions. Thanks to Dr. Pierre Bouchard for expert guidance on ITU recommendations. Thanks to Mathieu Châteauvert and many other CRC colleagues/students for feedback on using CRC-COVLIB.

Also thanks to Dr. Ivica Stevanovic. CRC-COVLIB’s implementation of the ITU-R P.1812 and ITU-R P.452 propagation models were tested using validation profiles and result files from Dr. Stevanovic.

Thanks to everyone at the NTIA who worked on developing propagation models and who contributed in making the source code available.

Thanks to the Lord my God, The Father, The Son & The Holy Spirit, for everything.

Martin-Pierre