Version

# Table of Contents

Table of contents

# CNOSSOS-EU Propagation Modules

## Public functions defined in the CnossosPropagation shared library

### Create, configure and delete the calculation engine

**CNOSSOS\_P2P\_CreateEngine**

**CNOSSOS\_P2P\_DeleteEngine**

**CNOSSOS\_P2P\_SelectMethod**

**CNOSSOS\_P2P\_GetMethod**

**CNOSSOS\_P2P\_GetVersion**

**CNOSSOS\_P2P\_SetOptions**

**CNOSSOS\_P2P\_GetOptions**

**CNOSSOS\_P2P\_SetMeteo**

**CNOSSOS\_P2P\_GetMeteo**

### Define the propagation path

**CNOSSOS\_P2P\_ClearPath**

**CNOSSOS\_P2P\_AddToPath**

**CNOSSOS\_P2P\_SetSoundPower**

### Create vertical extensions

**CNOSSOS\_P2P\_CreatePointSource**

**CNOSSOS\_P2P\_CreateLineSource**

**CNOSSOS\_P2P\_CreateReceiver**

**CNOSSOS\_P2P\_CreateBarrier**

**CNOSSOS\_P2P\_CreateVerticalWall**

**CNOSSOS\_P2P\_CreateVerticalEdge**

### Get the acoustical results

**CNOSSOS\_P2P\_GetResult**

**CNOSSOS\_P2P\_PrintPathData**

**CNOSSOS\_P2P\_PrintPathResults**

### Manage material properties

**CNOSSOS\_P2P\_GetMaterial**

**CNOSSOS\_P2P\_GetGValue**

**CNOSSOS\_P2P\_SetGValue**

**CNOSSOS\_P2P\_GetSigma**

**CNOSSOS\_P2P\_SetSigma**

**CNOSSOS\_P2P\_GetAlpha**

**CNOSSOS\_P2P\_SetAlpha**

### Utility functions

**CNOSSOS\_P2P\_GetVersionDLL**

**CNOSSOS\_P2P\_GetErrorMessage**

**CNOSSOS\_P2P\_ProcessPathFile**

# Class Index

## Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

**CNOSSOS\_P2P\_ENGINE (Hidden class for path calculation )** pagenum

**CNOSSOS\_P2P\_EXTENSION (Transparent pointer to vertical extension )** pagenum

**CNOSSOS\_P2P\_MATERIAL (Transparent pointer to material definition )** pagenum

**CNOSSOS\_P2P\_METEO (Meteorological input data )** pagenum

**CNOSSOS\_P2P\_OPTIONS (Options modifying the behavior of the calculation engine )** pagenum

# File Index

## File List

Here is a list of all files with brief descriptions:

**CnossosPropagation.cpp**

**CnossosPropagation.h (C-style API for the Cnossos-EU propagation modules )** pagenum

**dllmain.cpp**  pagenum

**stdafx.cpp**  pagenum

**stdafx.h**  pagenum

**targetver.h**  pagenum

# Class Documentation

## CNOSSOS\_P2P\_ENGINE Struct Reference

Hidden class for path calculation.

### Public Member Functions

**CNOSSOS\_P2P\_ENGINE** (const char \*name)

*constructor*

bool **selectMethod** (const char \*name)

*select the calculation method*

bool **doCalculation** (void)

*process the path*

### Public Attributes

ref\_ptr< CalculationMethod > **method**

*pointer to the calculation method*

PropagationPath **path**

*propagation path data*

ElementarySource **source**

PropagationPathOptions **options**

PathResult **result**

bool **calculationDone**

bool **hasErrors**

std::string **errorMessage**

### Detailed Description

Hidden class for path calculation.

### Constructor & Destructor Documentation

#### CNOSSOS\_P2P\_ENGINE::CNOSSOS\_P2P\_ENGINE (const char \**name*) [inline]

constructor

### Member Function Documentation

#### bool CNOSSOS\_P2P\_ENGINE::doCalculation (void) [inline]

process the path

#### bool CNOSSOS\_P2P\_ENGINE::selectMethod (const char \**name*) [inline]

select the calculation method

### Member Data Documentation

#### bool CNOSSOS\_P2P\_ENGINE::calculationDone

#### std::string CNOSSOS\_P2P\_ENGINE::errorMessage

#### bool CNOSSOS\_P2P\_ENGINE::hasErrors

#### ref\_ptr<CalculationMethod> CNOSSOS\_P2P\_ENGINE::method

pointer to the calculation method

#### PropagationPathOptions CNOSSOS\_P2P\_ENGINE::options

#### PropagationPath CNOSSOS\_P2P\_ENGINE::path

propagation path data

#### PathResult CNOSSOS\_P2P\_ENGINE::result

#### ElementarySource CNOSSOS\_P2P\_ENGINE::source

#### The documentation for this struct was generated from the following file:

**CnossosPropagation.cpp**

## CNOSSOS\_P2P\_EXTENSION Struct Reference

transparent pointer to vertical extension

### Detailed Description

transparent pointer to vertical extension

The documentation for this struct was generated from the following file:

**CnossosPropagation.cpp**

## CNOSSOS\_P2P\_MATERIAL Struct Reference

transparent pointer to material definition

### Detailed Description

transparent pointer to material definition

The documentation for this struct was generated from the following file:

**CnossosPropagation.cpp**

## CNOSSOS\_P2P\_METEO Struct Reference

Meteorological input data.

#include <CnossosPropagation.h>

### Public Attributes

**CNOSSOS\_P2P\_METEO\_MODEL** **model**

double **C0**

double **pFav**

double **temperature**

double **humidity**

### Detailed Description

Meteorological input data.

### Member Data Documentation

#### double CNOSSOS\_P2P\_METEO::C0

C0 correction term as defined in ISO 9613-2

#### double CNOSSOS\_P2P\_METEO::humidity

Air relative humidity (expressed as %)

#### CNOSSOS\_P2P\_METEO\_MODEL CNOSSOS\_P2P\_METEO::model

Choice of meteorological model

#### double CNOSSOS\_P2P\_METEO::pFav

Frequency of occurence of favourable propagation conditions

#### double CNOSSOS\_P2P\_METEO::temperature

Air temperature

#### The documentation for this struct was generated from the following file:

**CnossosPropagation.h**

## CNOSSOS\_P2P\_OPTIONS Struct Reference

Options modifying the behavior of the calculation engine.

#include <CnossosPropagation.h>

### Public Attributes

bool **CheckHorizontalAlignment**

*check validity of paths in the horizontal plane*

bool **CheckLateralDiffraction**

*check validity of laterally diffracted paths*

bool **CheckHeightLowerBound**

*check height of ray path with respect to upper limits of vertical obstacles*

bool **CheckHeightUpperBound**

*check height of ray path with respect to lower limits of vertical obstacles*

bool **CheckSourceSegment**

*check whether source line segments contains actual source position*

bool **CheckSoundPowerUnits**

*check conformity of sound power units and source geometry*

bool **ForceSourceToReceiver**

*reverse the path in case it is ordered from Receiver to Source*

bool **SimplifyPathGeometry**

*remove non representative terrain points*

bool **DisableReflections**

*disable reflections*

bool **DisableLateralDiffractions**

*disable lateral diffractions*

bool **LimitLateralDiffraction**

*disable reflections and vertical diffraction for laterally diffracted paths*

bool **ExcludeSoundPower**

*include sound power*

bool **ExcludeGeometricalSpread**

*include geometrical spread*

bool **ExcludeAirAbsorption**

*include atmospheric absorption*

### Detailed Description

Options modifying the behavior of the calculation engine.

### Member Data Documentation

#### bool CNOSSOS\_P2P\_OPTIONS::CheckHeightLowerBound

check height of ray path with respect to upper limits of vertical obstacles

#### bool CNOSSOS\_P2P\_OPTIONS::CheckHeightUpperBound

check height of ray path with respect to lower limits of vertical obstacles

#### bool CNOSSOS\_P2P\_OPTIONS::CheckHorizontalAlignment

check validity of paths in the horizontal plane

#### bool CNOSSOS\_P2P\_OPTIONS::CheckLateralDiffraction

check validity of laterally diffracted paths

#### bool CNOSSOS\_P2P\_OPTIONS::CheckSoundPowerUnits

check conformity of sound power units and source geometry

#### bool CNOSSOS\_P2P\_OPTIONS::CheckSourceSegment

check whether source line segments contains actual source position

#### bool CNOSSOS\_P2P\_OPTIONS::DisableLateralDiffractions

disable lateral diffractions

#### bool CNOSSOS\_P2P\_OPTIONS::DisableReflections

disable reflections

#### bool CNOSSOS\_P2P\_OPTIONS::ExcludeAirAbsorption

include atmospheric absorption

#### bool CNOSSOS\_P2P\_OPTIONS::ExcludeGeometricalSpread

include geometrical spread

#### bool CNOSSOS\_P2P\_OPTIONS::ExcludeSoundPower

include sound power

#### bool CNOSSOS\_P2P\_OPTIONS::ForceSourceToReceiver

reverse the path in case it is ordered from Receiver to Source

#### bool CNOSSOS\_P2P\_OPTIONS::LimitLateralDiffraction

disable reflections and vertical diffraction for laterally diffracted paths

#### bool CNOSSOS\_P2P\_OPTIONS::SimplifyPathGeometry

remove non representative terrain points

#### The documentation for this struct was generated from the following file:

**CnossosPropagation.h**

# File Documentation

## CnossosPropagation.cpp File Reference

#include "CnossosPropagation.h"

#include "PropagationPath.h"

#include "CalculationMethod.h"

#include "PathParseXML.h"

#include "PathResult.h"

#include <direct.h>

#include <string>

### Classes

struct **CNOSSOS\_P2P\_ENGINE**

*Hidden class for path calculation.* struct **CNOSSOS\_P2P\_MATERIAL**

*transparent pointer to material definition* struct **CNOSSOS\_P2P\_EXTENSION**

### *transparent pointer to vertical extension* Functions

char \* **CNOSSOS\_P2P\_GetVersionDLL** (void)

*Get the current version of the CnossosPropagation shared library.*

unsigned int **CNOSSOS\_P2P\_GetFreq** (double \*freq)

*Get the frequency range for the spectral data.*

**CNOSSOS\_P2P\_MATERIAL** \* **CNOSSOS\_P2P\_GetMaterial** (const char \*id, bool create\_if\_needed)

*Get material pointer.*

bool **CNOSSOS\_P2P\_GetGValue** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double &G)

*Get the ground factor for a given material.*

bool **CNOSSOS\_P2P\_SetGValue** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double const &G)

*Set the ground factor for a given material.*

bool **CNOSSOS\_P2P\_GetSigma** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double &sigma)

*Get the flow resistivity for a given material.*

bool **CNOSSOS\_P2P\_SetSigma** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double const &sigma)

*Set the flow resistivity for a given material.*

bool **CNOSSOS\_P2P\_GetAlpha** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double \*alpha)

*Get the spectral absorption factor for a given material.*

bool **CNOSSOS\_P2P\_SetAlpha** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double \*alpha)

**CNOSSOS\_P2P\_ENGINE** \* **CNOSSOS\_P2P\_CreateEngine** (const char \*name)

*Create a new calculation engine.*

bool **CNOSSOS\_P2P\_DeleteEngine** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Delete a calculation engine created by a call to* ***CNOSSOS\_P2P\_CreateEngine****.*

const char \* **CNOSSOS\_P2P\_GetMethod** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Get the name of the currently selected propagation path calculator.*

const char \* **CNOSSOS\_P2P\_GetVersion** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Get the version of the currently selected propagation path calculator.*

bool **CNOSSOS\_P2P\_SelectMethod** (**CNOSSOS\_P2P\_ENGINE** \*p2p, const char \*name)

*select the point to point method*

**CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreatePointSource** (double const &h)

*Create a vertical extension representing a point source.*

**CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateReceiver** (double const &h)

*Create a vertical extension representing a receiver point.*

**CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateBarrier** (double const &h, **CNOSSOS\_P2P\_MATERIAL** \*mat)

*Create a vertical extension for a thin barrier crossed by the propagation plane.*

**CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateVerticalWall** (double const &h, **CNOSSOS\_P2P\_MATERIAL** \*mat)

*Create an extension for a vertical wall acting as a reflector for the propagation path.*

**CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateVerticalEdge** (double const &h)

*Create an extension for a vertical edge causing lateral diffraction of the propagation path.*

**CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateLineSource** (double const &h, **CNOSSOS\_P2P\_POSITION** const segment[2], double const &fixedAngle)

*Create a vertical extension representing a line source segment.*

bool **CNOSSOS\_P2P\_ClearPath** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Clear the propagation path data.*

unsigned int **CNOSSOS\_P2P\_AddToPath** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_POSITION** const &pos, **CNOSSOS\_P2P\_MATERIAL** \*mat, **CNOSSOS\_P2P\_EXTENSION** \*ext)

*Add a new control point to the propagation path.*

unsigned int **CNOSSOS\_P2P\_GetResult** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_RESULT** index, double \*result)

*Get the acoustical results for the current propagation path.*

bool **CNOSSOS\_P2P\_SetSoundPower** (**CNOSSOS\_P2P\_ENGINE** \*p2p, double const \*Lw, **CNOSSOS\_P2P\_SPECTRUM\_WEIGHTING** weighting, **CNOSSOS\_P2P\_LW\_MEASUREMENT\_TYPE** type)

*Set the sound power associated with the source.*

const char \* **CNOSSOS\_P2P\_GetErrorMessage** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Get a description of the last error that occurred in the acoustical calculation.*

bool **CNOSSOS\_P2P\_PrintPathData** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Print the current propagation path to the application's default output device.*

bool **CNOSSOS\_P2P\_PrintPathResults** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Print an overview of the acoustical results to the application's output device.*

bool **CNOSSOS\_P2P\_SetOptions** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_OPTIONS** const &\_options)

*Set the current settings of the calculation options.*

bool **CNOSSOS\_P2P\_GetOptions** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_OPTIONS** &\_options)

*Get the current settings of the calculation options.*

bool **CNOSSOS\_P2P\_SetMeteo** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_METEO** const &\_meteo)

*Set the meteorological condition parameters.*

bool **CNOSSOS\_P2P\_GetMeteo** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_METEO** &\_meteo)

*Get the meteorological condition parameters.*

bool **CNOSSOS\_P2P\_ProcessPathFile** (**CNOSSOS\_P2P\_ENGINE** \*p2p, const char \*inputFile, const char \*outputFile)

*Process a XML input file and optionally produce an XML output file.*

### Function Documentation

#### unsigned int CNOSSOS\_P2P\_AddToPath (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_POSITION const &*pos*, CNOSSOS\_P2P\_MATERIAL \**mat*, CNOSSOS\_P2P\_EXTENSION \**ext* = 0)

Add a new control point to the propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |
| *pos* | Position of the control point |
| *mat* | Transparent pointer to a material as returned by a previous call to **CNOSSOS\_P2P\_GetMaterial** |
| *ext* | Transparent pointer to a vertical extension record as returned by a previous call to **CNOSSOS\_P2P\_CreatePointSource**, **CNOSSOS\_P2P\_CreateLineSource**, **CNOSSOS\_P2P\_CreateReceiver**, **CNOSSOS\_P2P\_CreateBarrier**, **CNOSSOS\_P2P\_CreateVerticalWall** or **CNOSSOS\_P2P\_CreateVerticalEdge**. For intermediate boundary positions without vertical extensions, this argument is set to NULL. |

##### Returns:

The number of positions in the propagation path after appending the specified control point

##### Note:

A control point is part of the boundary separating the solid 2.5D model from the air above. Control points can be located on the ground or on top of man-made obstacles. Altitude of control points is measured against a geo-spatial reference coordinate system. Vertical extensions encode information related to vertical objects extending above the control point's position. The height of a vertical extension is measured relative to the altitude of the control point.

A valid path should have exactly one source and one receiver vertical extension at its extreme points. Paths may start at the source and end with a receiver or vice-versa.

#### bool CNOSSOS\_P2P\_ClearPath (CNOSSOS\_P2P\_ENGINE \**p2p*)

Clear the propagation path data.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

true if the call succeeded, false otherwise

##### Note:

A propagation path is created by successive calls to **CNOSSOS\_P2P\_AddToPath**. A call to **CNOSSOS\_P2P\_GetResult** will close the path and trigger the acoustical calculations. In order to restart the calculation for a different path, applications must call the **CNOSSOS\_P2P\_ClearPath** function before constructing a new one.

##### Returns:

true if the call succeeded, false otherwise

#### CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateBarrier (double const &*h*, CNOSSOS\_P2P\_MATERIAL \**mat* = 0)

Create a vertical extension for a thin barrier crossed by the propagation plane.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the barrier above the boundary profile |
| *mat* | Pointer to the material covering the barrier, as returned by a call to **CNOSSOS\_P2P\_GetMaterial** |

##### Returns:

A transparent pointer to a vertical extension record

#### CNOSSOS\_P2P\_ENGINE\* CNOSSOS\_P2P\_CreateEngine (const char \**name* = 0)

Create a new calculation engine.

##### Parameters:

|  |  |
| --- | --- |
| *name* | If specified, initializes the engine for the specified calculation method |

##### See also:

**CNOSSOS\_P2P\_SelectMethod**

##### Returns:

An opaque pointer to the newly created calculation engine

#### CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateLineSource (double const &*h*, CNOSSOS\_P2P\_POSITION const*segment*[2], double const &*fixedAngle* = 0.0)

Create a vertical extension representing a line source segment.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the source above the boundary profile |
| *segment* | end-points of the source line segment as projected on the ground |
| *fixedAngle* | explicit angle of view of the segment as seen from the receiver. |

##### Returns:

A transparent pointer to a vertical extension record

##### Note:

Setting a fixed angle is necessary in case (inverse) ray-tracing is used for constructing the propagation path as each ray is considered representative of an implicit sector of propagation with fixed opening angle and having the propagation path as its bisector. If the fixed angle argument is set to zero, the calculation will explicitly calculate the angle of view of the segment as seen from the source, which is typically the case when using beam-tracing or image-source techniques for constructing the propagation paths.

#### CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreatePointSource (double const &*h*)

Create a vertical extension representing a point source.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the point source above the boundary profile |

##### Returns:

A transparent pointer to a vertical extension record

#### CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateReceiver (double const &*h*)

Create a vertical extension representing a receiver point.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the receiver above the boundary profile |

##### Returns:

A transparent pointer to a vertical extension record

#### CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateVerticalEdge (double const &*h*)

Create an extension for a vertical edge causing lateral diffraction of the propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the diffracting edge above the boundary profile |

##### Returns:

A transparent pointer to a vertical extension record

#### CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateVerticalWall (double const &*h*, CNOSSOS\_P2P\_MATERIAL \**mat* = 0)

Create an extension for a vertical wall acting as a reflector for the propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the reflecting wall above the boundary profile |
| *mat* | Pointer to the material covering the reflecting surface, as returned by a call to **CNOSSOS\_P2P\_GetMaterial** |

##### Returns:

A transparent pointer to a vertical extension record

#### bool CNOSSOS\_P2P\_DeleteEngine (CNOSSOS\_P2P\_ENGINE \**p2p*)

Delete a calculation engine created by a call to **CNOSSOS\_P2P\_CreateEngine**.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_GetAlpha (CNOSSOS\_P2P\_MATERIAL \**mat*, double \**alpha*)

Get the spectral absorption factor for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *alpha* | Array containing the values of the absorption factor spectrum |

##### Returns:

true if the call succeeded, false otherwise

#### const char\* CNOSSOS\_P2P\_GetErrorMessage (CNOSSOS\_P2P\_ENGINE \**p2p*)

Get a description of the last error that occurred in the acoustical calculation.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |

##### Returns:

Pointer to a string containing the error message.

#### unsigned int CNOSSOS\_P2P\_GetFreq (double \**freq*)

Get the frequency range for the spectral data.

##### Parameters:

|  |  |
| --- | --- |
| *freq* | Array in which to return the frequency bands associated with spectral information passed to and from the shared library. If a NULL pointer is specified, the function returns the number of values that would have been written on output but does not transfer any actual values. |

##### Returns:

The number of frequency bands.

##### Note:

It is up to the caller to pass in an array of sufficient size. Applications can determine the minimal size of this array by first calling the function with a NULL argument.

#### bool CNOSSOS\_P2P\_GetGValue (CNOSSOS\_P2P\_MATERIAL \**mat*, double &*G*)

Get the ground factor for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *G* | New value of the ground factor |

##### Returns:

true if the call succeeded, false otherwise

#### CNOSSOS\_P2P\_MATERIAL\* CNOSSOS\_P2P\_GetMaterial (const char \**id*, bool*create\_if\_needed* = false)

Get material pointer.

##### Parameters:

|  |  |
| --- | --- |
| *id* | Unique textual identifier for the material |
| *create\_if\_needed* | If a material with the given name does not exist and this flag is set to true, the library will create a new material; otherwise this function returns a NULL pointer. |

##### Returns:

An opaque pointer to the material or NULL if no material with the given name exists and the create\_if\_needed flag is set to false.

#### bool CNOSSOS\_P2P\_GetMeteo (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_METEO &*meteo*)

Get the meteorological condition parameters.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *meteo* | Data structure used to return the current settings of the meteorological parameters |

##### Returns:

true if the call succeeded, false otherwise

#### const char\* CNOSSOS\_P2P\_GetMethod (CNOSSOS\_P2P\_ENGINE \**p2p*)

Get the name of the currently selected propagation path calculator.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

Pointer to a string containing the name of the method

#### bool CNOSSOS\_P2P\_GetOptions (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_OPTIONS &*options*)

Get the current settings of the calculation options.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *options* | Data structure used to return the current settings. |

##### Returns:

true if the call succeeded, false otherwise

#### unsigned int CNOSSOS\_P2P\_GetResult (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_RESULT*index*, double \**result*)

Get the acoustical results for the current propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *index* | Select the type of results to be returned |
| *result* | Array in which to return the acoustical results. If a NULL pointer is specified, the function returns the number of values that would have been written on output but does not transfer any actual values. |

##### Returns:

The number of values written (or not) to the array of results or zero in case an error occurred in the calculation.

##### Note:

It is up to the caller to pass in an array of sufficient size. Applications can determine the minimal size of this array by first calling the function with a NULL argument.

#### bool CNOSSOS\_P2P\_GetSigma (CNOSSOS\_P2P\_MATERIAL \**mat*, double &*sigma*)

Get the flow resistivity for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *sigma* | Value of the flow resistivity (in kPa.s/m²) |

##### Returns:

true if the call succeeded, false otherwise

#### const char\* CNOSSOS\_P2P\_GetVersion (CNOSSOS\_P2P\_ENGINE \**p2p*)

Get the version of the currently selected propagation path calculator.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

Pointer to a string containing the current version of the method

#### char\* CNOSSOS\_P2P\_GetVersionDLL (void)

Get the current version of the CnossosPropagation shared library.

##### Returns:

String encoded version of the shared library.

#### bool CNOSSOS\_P2P\_PrintPathData (CNOSSOS\_P2P\_ENGINE \**p2p*)

Print the current propagation path to the application's default output device.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_PrintPathResults (CNOSSOS\_P2P\_ENGINE \**p2p*)

Print an overview of the acoustical results to the application's output device.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_ProcessPathFile (CNOSSOS\_P2P\_ENGINE \**p2p*, const char \**inputFile*, const char \**outputFile* = 0)

Process a XML input file and optionally produce an XML output file.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *inputFile* | Name of an XML input file |
| *outputFile* | Name of an (optional) XML output file |

##### Returns:

true if the call succeeded, false otherwise

##### Note:

The name of the input file can include a full path specification or be a path name relative to the current working directory. The name of the output file can be either a full path specification or a path name relative to the location of the input file. Even if no output file is generated, the application can use the **CNOSSOS\_P2P\_GetResult** function to access the acoustical results associated with the path defined in the input file.

#### bool CNOSSOS\_P2P\_SelectMethod (CNOSSOS\_P2P\_ENGINE \**p2p*, const char \**name*)

select the point to point method

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |
| *name* | Textual identifier of the calculation method. Valid methods include "ISO-9613-2", "JRC-DRAFT-2010" and "JRC-2012" |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_SetAlpha (CNOSSOS\_P2P\_MATERIAL \**mat*, double \**alpha*)

#### bool CNOSSOS\_P2P\_SetGValue (CNOSSOS\_P2P\_MATERIAL \**mat*, double const &*G*)

Set the ground factor for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *G* | New value of the ground factor |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_SetMeteo (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_METEO const &*meteo*)

Set the meteorological condition parameters.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *meteo* | Data structure containing the new values of the meteorological parameters. |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_SetOptions (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_OPTIONS const &*options*)

Set the current settings of the calculation options.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *options* | Data structure containing the new values of the options. |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_SetSigma (CNOSSOS\_P2P\_MATERIAL \**mat*, double const &*sigma*)

Set the flow resistivity for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *sigma* | New value of the flow resistivity (in kPa.s/m²) |

##### Returns:

true if the call succeeded, false otherwise

#### bool CNOSSOS\_P2P\_SetSoundPower (CNOSSOS\_P2P\_ENGINE \**p2p*, double const \**Lw*, CNOSSOS\_P2P\_SPECTRUM\_WEIGHTING*spectrumWeighting* = CNOSSOS\_P2P\_SPECTRUM\_LIN, CNOSSOS\_P2P\_LW\_MEASUREMENT\_TYPE*measurementType* = CNOSSOS\_P2P\_LW\_UNDEFINED)

Set the sound power associated with the source.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *Lw* | Pointer to an array containing the sound power spectrum. |
| *spectrumWeighting* | Weighting of the sound power spectrum |
| *measurementType* | Measurement conditions for the sound power spectrum |

##### Returns:

true if the call succeeded, false otherwise

## CnossosPropagation.h File Reference

C-style API for the Cnossos-EU propagation modules.

### Classes

struct **CNOSSOS\_P2P\_METEO**

*Meteorological input data.* struct **CNOSSOS\_P2P\_OPTIONS**

### *Options modifying the behavior of the calculation engine.* Defines

#define **\_CNOSSOS\_DLL\_DECL\_**  extern "C" \_\_declspec(dllimport)

#define **CNOSSOS\_P2P\_VERSION\_API**  "1.001"

*current version associated with this API*

### Typedefs

typedef double **CNOSSOS\_P2P\_POSITION** [3]

*Encoding of positions using 3 dimensional (X,Y,Z) coordinates.*

### Enumerations

enum **CNOSSOS\_P2P\_SPECTRUM\_WEIGHTING** { **CNOSSOS\_P2P\_SPECTRUM\_LIN** = 0, **CNOSSOS\_P2P\_SPECTRUM\_dBA** = 1 }

*Sound power frequency weighting.* enum **CNOSSOS\_P2P\_LW\_MEASUREMENT\_TYPE** { **CNOSSOS\_P2P\_LW\_UNDEFINED** = 0, **CNOSSOS\_P2P\_LW\_FREEFIELD** = 1, **CNOSSOS\_P2P\_LW\_HEMISPHERICAL** = 2 }

*Sound power measurement conditions.* enum **CNOSSOS\_P2P\_METEO\_MODEL** { **CNOSSOS\_P2P\_METEO\_DEFAULT** = 0, **CNOSSOS\_P2P\_METEO\_ISO9613** = 1, **CNOSSOS\_P2P\_METEO\_JRC2012** = 2 }

*Choice of meteorological averaging.* enum **CNOSSOS\_P2P\_RESULT** { **CNOSSOS\_P2P\_RESULT\_LP\_AVG\_dBA** = -1, **CNOSSOS\_P2P\_RESULT\_LP\_FAV\_dBA** = -2, **CNOSSOS\_P2P\_RESULT\_LP\_HOM\_dBA** = -3, **CNOSSOS\_P2P\_RESULT\_LP\_AVG** = 0, **CNOSSOS\_P2P\_RESULT\_LP\_FAV** = 1, **CNOSSOS\_P2P\_RESULT\_LP\_HOM** = 2, **CNOSSOS\_P2P\_RESULT\_ATT\_FAV** = 3, **CNOSSOS\_P2P\_RESULT\_ATT\_HOM** = 4, **CNOSSOS\_P2P\_RESULT\_LW\_SOURCE** = 5, **CNOSSOS\_P2P\_RESULT\_DELTA\_LW** = 6, **CNOSSOS\_P2P\_RESULT\_ATT\_GEO** = 7, **CNOSSOS\_P2P\_RESULT\_ATT\_ATM** = 8, **CNOSSOS\_P2P\_RESULT\_ATT\_REF** = 9, **CNOSSOS\_P2P\_RESULT\_ATT\_DIF** = 10, **CNOSSOS\_P2P\_RESULT\_ATT\_SIZE** = 11 }

### *Select the result to be returned by the call to CNOSSOS\_P2P\_GetResult.* Functions

\_CNOSSOS\_DLL\_DECL\_ char \* **CNOSSOS\_P2P\_GetVersionDLL** (void)

*Get the current version of the CnossosPropagation shared library.*

\_CNOSSOS\_DLL\_DECL\_ unsigned int **CNOSSOS\_P2P\_GetFreq** (double \*freq)

*Get the frequency range for the spectral data.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_MATERIAL** \* **CNOSSOS\_P2P\_GetMaterial** (const char \*id, bool create\_if\_needed=false)

*Get material pointer.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_SetSigma** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double const &sigma)

*Set the flow resistivity for a given material.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_GetSigma** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double &sigma)

*Get the flow resistivity for a given material.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_SetGValue** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double const &G)

*Set the ground factor for a given material.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_GetGValue** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double &G)

*Get the ground factor for a given material.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_SetAlpha** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double const \*alpha)

*Set the spectral absorption factor for a given material.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_GetAlpha** (**CNOSSOS\_P2P\_MATERIAL** \*mat, double \*alpha)

*Get the spectral absorption factor for a given material.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_ENGINE** \* **CNOSSOS\_P2P\_CreateEngine** (const char \*name=0)

*Create a new calculation engine.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_DeleteEngine** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Delete a calculation engine created by a call to* ***CNOSSOS\_P2P\_CreateEngine****.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_SelectMethod** (**CNOSSOS\_P2P\_ENGINE** \*p2p, const char \*name)

*select the point to point method*

\_CNOSSOS\_DLL\_DECL\_ const char \* **CNOSSOS\_P2P\_GetMethod** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Get the name of the currently selected propagation path calculator.*

\_CNOSSOS\_DLL\_DECL\_ const char \* **CNOSSOS\_P2P\_GetVersion** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Get the version of the currently selected propagation path calculator.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_ClearPath** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Clear the propagation path data.*

\_CNOSSOS\_DLL\_DECL\_ unsigned int **CNOSSOS\_P2P\_AddToPath** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_POSITION** const &pos, **CNOSSOS\_P2P\_MATERIAL** \*mat, **CNOSSOS\_P2P\_EXTENSION** \*ext=0)

*Add a new control point to the propagation path.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreatePointSource** (double const &h)

*Create a vertical extension representing a point source.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateReceiver** (double const &h)

*Create a vertical extension representing a receiver point.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateBarrier** (double const &h, **CNOSSOS\_P2P\_MATERIAL** \*mat=0)

*Create a vertical extension for a thin barrier crossed by the propagation plane.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateVerticalWall** (double const &h, **CNOSSOS\_P2P\_MATERIAL** \*mat=0)

*Create an extension for a vertical wall acting as a reflector for the propagation path.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateVerticalEdge** (double const &h)

*Create an extension for a vertical edge causing lateral diffraction of the propagation path.*

\_CNOSSOS\_DLL\_DECL\_ **CNOSSOS\_P2P\_EXTENSION** \* **CNOSSOS\_P2P\_CreateLineSource** (double const &h, **CNOSSOS\_P2P\_POSITION** const segment[2], double const &fixedAngle=0.0)

*Create a vertical extension representing a line source segment.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_SetSoundPower** (**CNOSSOS\_P2P\_ENGINE** \*p2p, double const \*Lw, **CNOSSOS\_P2P\_SPECTRUM\_WEIGHTING** spectrumWeighting=CNOSSOS\_P2P\_SPECTRUM\_LIN, **CNOSSOS\_P2P\_LW\_MEASUREMENT\_TYPE** measurementType=CNOSSOS\_P2P\_LW\_UNDEFINED)

*Set the sound power associated with the source.*

\_CNOSSOS\_DLL\_DECL\_ unsigned int **CNOSSOS\_P2P\_GetResult** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_RESULT** index, double \*result)

*Get the acoustical results for the current propagation path.*

\_CNOSSOS\_DLL\_DECL\_ const char \* **CNOSSOS\_P2P\_GetErrorMessage** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Get a description of the last error that occurred in the acoustical calculation.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_PrintPathData** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Print the current propagation path to the application's default output device.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_PrintPathResults** (**CNOSSOS\_P2P\_ENGINE** \*p2p)

*Print an overview of the acoustical results to the application's output device.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_GetOptions** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_OPTIONS** &options)

*Get the current settings of the calculation options.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_SetOptions** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_OPTIONS** const &options)

*Set the current settings of the calculation options.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_SetMeteo** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_METEO** const &meteo)

*Set the meteorological condition parameters.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_GetMeteo** (**CNOSSOS\_P2P\_ENGINE** \*p2p, **CNOSSOS\_P2P\_METEO** &meteo)

*Get the meteorological condition parameters.*

\_CNOSSOS\_DLL\_DECL\_ bool **CNOSSOS\_P2P\_ProcessPathFile** (**CNOSSOS\_P2P\_ENGINE** \*p2p, const char \*inputFile, const char \*outputFile=0)

*Process a XML input file and optionally produce an XML output file.*

### Detailed Description

C-style API for the Cnossos-EU propagation modules.

##### Author:

dirk.van-maercke@cstb.fr

### Define Documentation

#### #define \_CNOSSOS\_DLL\_DECL\_  extern "C" \_\_declspec(dllimport)

#### #define CNOSSOS\_P2P\_VERSION\_API  "1.001"

current version associated with this API

### Typedef Documentation

#### typedef double CNOSSOS\_P2P\_POSITION[3]

Encoding of positions using 3 dimensional (X,Y,Z) coordinates.

### Enumeration Type Documentation

#### enum CNOSSOS\_P2P\_LW\_MEASUREMENT\_TYPE

Sound power measurement conditions.

**Enumerator:**

***CNOSSOS\_P2P\_LW\_UNDEFINED*** Measurement conditions undefined

***CNOSSOS\_P2P\_LW\_FREEFIELD*** Sound power measured under free field conditions

***CNOSSOS\_P2P\_LW\_HEMISPHERICAL*** Sound power measured under hemispherical conditions

#### enum CNOSSOS\_P2P\_METEO\_MODEL

Choice of meteorological averaging.

**Enumerator:**

***CNOSSOS\_P2P\_METEO\_DEFAULT*** Use default model corresponding to the selected calculation method

***CNOSSOS\_P2P\_METEO\_ISO9613*** Use the C0 correction as defined in the ISO 9613-2 standard

***CNOSSOS\_P2P\_METEO\_JRC2012*** Use the frequency of occurence of favorable propagation conditions

#### enum CNOSSOS\_P2P\_RESULT

Select the result to be returned by the call to **CNOSSOS\_P2P\_GetResult**.

**Enumerator:**

***CNOSSOS\_P2P\_RESULT\_LP\_AVG\_dBA*** global, dB(A) weighted, long term averaged noise level

***CNOSSOS\_P2P\_RESULT\_LP\_FAV\_dBA*** global, dB(A) weighted, noise level under favorable propagation conditions

***CNOSSOS\_P2P\_RESULT\_LP\_HOM\_dBA*** global, dB(A) weighted, noise level under homogeneous propagation conditions

***CNOSSOS\_P2P\_RESULT\_LP\_AVG*** long-time averaged noise level spectrum

***CNOSSOS\_P2P\_RESULT\_LP\_FAV*** noise level spectrum under favorable propagation conditions

***CNOSSOS\_P2P\_RESULT\_LP\_HOM*** noise level spectrum under homogeneous propagation conditions

***CNOSSOS\_P2P\_RESULT\_ATT\_FAV*** excess attenuation spectrum under favorable propagation conditions

***CNOSSOS\_P2P\_RESULT\_ATT\_HOM*** excess attenuation spectrum under homogeneous propagation conditions

***CNOSSOS\_P2P\_RESULT\_LW\_SOURCE*** sound power spectrum

***CNOSSOS\_P2P\_RESULT\_DELTA\_LW*** sound power conversion, free field versus hemispherical radiation conditions

***CNOSSOS\_P2P\_RESULT\_ATT\_GEO*** attenuation due to geometrical spreading

***CNOSSOS\_P2P\_RESULT\_ATT\_ATM*** attenuation due to atmospheric absorption

***CNOSSOS\_P2P\_RESULT\_ATT\_REF*** attenuation due to reflections by vertical obstacles

***CNOSSOS\_P2P\_RESULT\_ATT\_DIF*** attenuation due to lateral diffraction by vertical edges

***CNOSSOS\_P2P\_RESULT\_ATT\_SIZE*** attenuation due to finite size of vertical obstacles

#### enum CNOSSOS\_P2P\_SPECTRUM\_WEIGHTING

Sound power frequency weighting.

**Enumerator:**

***CNOSSOS\_P2P\_SPECTRUM\_LIN*** Spectrum is unweighted

***CNOSSOS\_P2P\_SPECTRUM\_dBA*** Spectrum is A-weighted

### Function Documentation

#### \_CNOSSOS\_DLL\_DECL\_ unsigned int CNOSSOS\_P2P\_AddToPath (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_POSITION const &*pos*, CNOSSOS\_P2P\_MATERIAL \**mat*, CNOSSOS\_P2P\_EXTENSION \**ext* = 0)

Add a new control point to the propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |
| *pos* | Position of the control point |
| *mat* | Transparent pointer to a material as returned by a previous call to **CNOSSOS\_P2P\_GetMaterial** |
| *ext* | Transparent pointer to a vertical extension record as returned by a previous call to **CNOSSOS\_P2P\_CreatePointSource**, **CNOSSOS\_P2P\_CreateLineSource**, **CNOSSOS\_P2P\_CreateReceiver**, **CNOSSOS\_P2P\_CreateBarrier**, **CNOSSOS\_P2P\_CreateVerticalWall** or **CNOSSOS\_P2P\_CreateVerticalEdge**. For intermediate boundary positions without vertical extensions, this argument is set to NULL. |

##### Returns:

The number of positions in the propagation path after appending the specified control point

##### Note:

A control point is part of the boundary separating the solid 2.5D model from the air above. Control points can be located on the ground or on top of man-made obstacles. Altitude of control points is measured against a geo-spatial reference coordinate system. Vertical extensions encode information related to vertical objects extending above the control point's position. The height of a vertical extension is measured relative to the altitude of the control point.

A valid path should have exactly one source and one receiver vertical extension at its extreme points. Paths may start at the source and end with a receiver or vice-versa.

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_ClearPath (CNOSSOS\_P2P\_ENGINE \**p2p*)

Clear the propagation path data.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

true if the call succeeded, false otherwise

##### Note:

A propagation path is created by successive calls to **CNOSSOS\_P2P\_AddToPath**. A call to **CNOSSOS\_P2P\_GetResult** will close the path and trigger the acoustical calculations. In order to restart the calculation for a different path, applications must call the **CNOSSOS\_P2P\_ClearPath** function before constructing a new one.

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateBarrier (double const &*h*, CNOSSOS\_P2P\_MATERIAL \**mat* = 0)

Create a vertical extension for a thin barrier crossed by the propagation plane.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the barrier above the boundary profile |
| *mat* | Pointer to the material covering the barrier, as returned by a call to **CNOSSOS\_P2P\_GetMaterial** |

##### Returns:

A transparent pointer to a vertical extension record

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_ENGINE\* CNOSSOS\_P2P\_CreateEngine (const char \**name* = 0)

Create a new calculation engine.

##### Parameters:

|  |  |
| --- | --- |
| *name* | If specified, initializes the engine for the specified calculation method |

##### See also:

**CNOSSOS\_P2P\_SelectMethod**

##### Returns:

An opaque pointer to the newly created calculation engine

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateLineSource (double const &*h*, CNOSSOS\_P2P\_POSITION const*segment*[2], double const &*fixedAngle* = 0.0)

Create a vertical extension representing a line source segment.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the source above the boundary profile |
| *segment* | end-points of the source line segment as projected on the ground |
| *fixedAngle* | explicit angle of view of the segment as seen from the receiver. |

##### Returns:

A transparent pointer to a vertical extension record

##### Note:

Setting a fixed angle is necessary in case (inverse) ray-tracing is used for constructing the propagation path as each ray is considered representative of an implicit sector of propagation with fixed opening angle and having the propagation path as its bisector. If the fixed angle argument is set to zero, the calculation will explicitly calculate the angle of view of the segment as seen from the source, which is typically the case when using beam-tracing or image-source techniques for constructing the propagation paths.

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreatePointSource (double const &*h*)

Create a vertical extension representing a point source.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the point source above the boundary profile |

##### Returns:

A transparent pointer to a vertical extension record

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateReceiver (double const &*h*)

Create a vertical extension representing a receiver point.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the receiver above the boundary profile |

##### Returns:

A transparent pointer to a vertical extension record

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateVerticalEdge (double const &*h*)

Create an extension for a vertical edge causing lateral diffraction of the propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the diffracting edge above the boundary profile |

##### Returns:

A transparent pointer to a vertical extension record

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_EXTENSION\* CNOSSOS\_P2P\_CreateVerticalWall (double const &*h*, CNOSSOS\_P2P\_MATERIAL \**mat* = 0)

Create an extension for a vertical wall acting as a reflector for the propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *h* | height of the reflecting wall above the boundary profile |
| *mat* | Pointer to the material covering the reflecting surface, as returned by a call to **CNOSSOS\_P2P\_GetMaterial** |

##### Returns:

A transparent pointer to a vertical extension record

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_DeleteEngine (CNOSSOS\_P2P\_ENGINE \**p2p*)

Delete a calculation engine created by a call to **CNOSSOS\_P2P\_CreateEngine**.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_GetAlpha (CNOSSOS\_P2P\_MATERIAL \**mat*, double \**alpha*)

Get the spectral absorption factor for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *alpha* | Array containing the values of the absorption factor spectrum |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ const char\* CNOSSOS\_P2P\_GetErrorMessage (CNOSSOS\_P2P\_ENGINE \**p2p*)

Get a description of the last error that occurred in the acoustical calculation.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |

##### Returns:

Pointer to a string containing the error message.

#### \_CNOSSOS\_DLL\_DECL\_ unsigned int CNOSSOS\_P2P\_GetFreq (double \**freq*)

Get the frequency range for the spectral data.

##### Parameters:

|  |  |
| --- | --- |
| *freq* | Array in which to return the frequency bands associated with spectral information passed to and from the shared library. If a NULL pointer is specified, the function returns the number of values that would have been written on output but does not transfer any actual values. |

##### Returns:

The number of frequency bands.

##### Note:

It is up to the caller to pass in an array of sufficient size. Applications can determine the minimal size of this array by first calling the function with a NULL argument.

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_GetGValue (CNOSSOS\_P2P\_MATERIAL \**mat*, double &*G*)

Get the ground factor for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *G* | New value of the ground factor |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ CNOSSOS\_P2P\_MATERIAL\* CNOSSOS\_P2P\_GetMaterial (const char \**id*, bool*create\_if\_needed* = false)

Get material pointer.

##### Parameters:

|  |  |
| --- | --- |
| *id* | Unique textual identifier for the material |
| *create\_if\_needed* | If a material with the given name does not exist and this flag is set to true, the library will create a new material; otherwise this function returns a NULL pointer. |

##### Returns:

An opaque pointer to the material or NULL if no material with the given name exists and the create\_if\_needed flag is set to false.

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_GetMeteo (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_METEO &*meteo*)

Get the meteorological condition parameters.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *meteo* | Data structure used to return the current settings of the meteorological parameters |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ const char\* CNOSSOS\_P2P\_GetMethod (CNOSSOS\_P2P\_ENGINE \**p2p*)

Get the name of the currently selected propagation path calculator.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

Pointer to a string containing the name of the method

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_GetOptions (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_OPTIONS &*options*)

Get the current settings of the calculation options.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *options* | Data structure used to return the current settings. |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ unsigned int CNOSSOS\_P2P\_GetResult (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_RESULT*index*, double \**result*)

Get the acoustical results for the current propagation path.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *index* | Select the type of results to be returned |
| *result* | Array in which to return the acoustical results. If a NULL pointer is specified, the function returns the number of values that would have been written on output but does not transfer any actual values. |

##### Returns:

The number of values written (or not) to the array of results or zero in case an error occurred in the calculation.

##### Note:

It is up to the caller to pass in an array of sufficient size. Applications can determine the minimal size of this array by first calling the function with a NULL argument.

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_GetSigma (CNOSSOS\_P2P\_MATERIAL \**mat*, double &*sigma*)

Get the flow resistivity for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *sigma* | Value of the flow resistivity (in kPa.s/m²) |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ const char\* CNOSSOS\_P2P\_GetVersion (CNOSSOS\_P2P\_ENGINE \**p2p*)

Get the version of the currently selected propagation path calculator.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |

##### Returns:

Pointer to a string containing the current version of the method

#### \_CNOSSOS\_DLL\_DECL\_ char\* CNOSSOS\_P2P\_GetVersionDLL (void)

Get the current version of the CnossosPropagation shared library.

##### Returns:

String encoded version of the shared library.

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_PrintPathData (CNOSSOS\_P2P\_ENGINE \**p2p*)

Print the current propagation path to the application's default output device.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_PrintPathResults (CNOSSOS\_P2P\_ENGINE \**p2p*)

Print an overview of the acoustical results to the application's output device.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_ProcessPathFile (CNOSSOS\_P2P\_ENGINE \**p2p*, const char \**inputFile*, const char \**outputFile* = 0)

Process a XML input file and optionally produce an XML output file.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *inputFile* | Name of an XML input file |
| *outputFile* | Name of an (optional) XML output file |

##### Returns:

true if the call succeeded, false otherwise

##### Note:

The name of the input file can include a full path specification or be a path name relative to the current working directory. The name of the output file can be either a full path specification or a path name relative to the location of the input file. Even if no output file is generated, the application can use the **CNOSSOS\_P2P\_GetResult** function to access the acoustical results associated with the path defined in the input file.

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_SelectMethod (CNOSSOS\_P2P\_ENGINE \**p2p*, const char \**name*)

select the point to point method

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine |
| *name* | Textual identifier of the calculation method. Valid methods include "ISO-9613-2", "JRC-DRAFT-2010" and "JRC-2012" |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_SetAlpha (CNOSSOS\_P2P\_MATERIAL \**mat*, double const \**alpha*)

Set the spectral absorption factor for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *alpha* | Array containing the values of the absorption factor spectrum |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_SetGValue (CNOSSOS\_P2P\_MATERIAL \**mat*, double const &*G*)

Set the ground factor for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *G* | New value of the ground factor |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_SetMeteo (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_METEO const &*meteo*)

Set the meteorological condition parameters.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *meteo* | Data structure containing the new values of the meteorological parameters. |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_SetOptions (CNOSSOS\_P2P\_ENGINE \**p2p*, CNOSSOS\_P2P\_OPTIONS const &*options*)

Set the current settings of the calculation options.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *options* | Data structure containing the new values of the options. |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_SetSigma (CNOSSOS\_P2P\_MATERIAL \**mat*, double const &*sigma*)

Set the flow resistivity for a given material.

##### Parameters:

|  |  |
| --- | --- |
| *mat* | An opaque pointer returned by a call to **CNOSSOS\_P2P\_GetMaterial** |
| *sigma* | New value of the flow resistivity (in kPa.s/m²) |

##### Returns:

true if the call succeeded, false otherwise

#### \_CNOSSOS\_DLL\_DECL\_ bool CNOSSOS\_P2P\_SetSoundPower (CNOSSOS\_P2P\_ENGINE \**p2p*, double const \**Lw*, CNOSSOS\_P2P\_SPECTRUM\_WEIGHTING*spectrumWeighting* = CNOSSOS\_P2P\_SPECTRUM\_LIN, CNOSSOS\_P2P\_LW\_MEASUREMENT\_TYPE*measurementType* = CNOSSOS\_P2P\_LW\_UNDEFINED)

Set the sound power associated with the source.

##### Parameters:

|  |  |
| --- | --- |
| *p2p* | An opaque pointer returned by a previous call to CNOSSOS\_P2P\_CreateEngine. |
| *Lw* | Pointer to an array containing the sound power spectrum. |
| *spectrumWeighting* | Weighting of the sound power spectrum |
| *measurementType* | Measurement conditions for the sound power spectrum |

##### Returns:

true if the call succeeded, false otherwise

## dllmain.cpp File Reference

#include "stdafx.h"

### Functions

BOOL APIENTRY **DllMain** (HMODULE hModule, DWORD ul\_reason\_for\_call, LPVOID lpReserved)

### Function Documentation

#### BOOL APIENTRY DllMain (HMODULE*hModule*, DWORD*ul\_reason\_for\_call*, LPVOID*lpReserved*)

#### 

## doc.dox File Reference

## stdafx.cpp File Reference

#include "stdafx.h"

## stdafx.h File Reference

#include "targetver.h"

#include <windows.h>

### Defines

#define **WIN32\_LEAN\_AND\_MEAN**

### Define Documentation

#### #define WIN32\_LEAN\_AND\_MEAN

#### 

## targetver.h File Reference

#include <SDKDDKVer.h>

# Index

INDEX