

**rl\_raylib**  
1.1.7

Generated by Doxygen 1.7.1

Wed Aug 11 2010 14:12:18

## Contents

<b>1</b>	<b>rl_RayLib User's Guide</b>	<b>1</b>
1.1	Copyright and License . . . . .	1
1.2	Purpose . . . . .	1
1.2.1	Ray class . . . . .	2
1.2.2	Reflectivity classes . . . . .	2
<b>2</b>	<b>Directory Hierarchy</b>	<b>2</b>
2.1	Directories . . . . .	2
<b>3</b>	<b>Class Index</b>	<b>3</b>
3.1	Class List . . . . .	3
<b>4</b>	<b>Directory Documentation</b>	<b>4</b>
4.1	rl_raylib/ Directory Reference . . . . .	4
<b>5</b>	<b>Class Documentation</b>	<b>5</b>
5.1	rl_DielectricData Class Reference . . . . .	5
5.1.1	Detailed Description . . . . .	6
5.1.2	Constructor & Destructor Documentation . . . . .	6
5.1.3	Member Function Documentation . . . . .	7
5.2	rl_DielectricData::rl_DielectricDataBinPOD Struct Reference . . . . .	9
5.2.1	Detailed Description . . . . .	10
5.2.2	Member Data Documentation . . . . .	10
5.3	rl_DielectricLayer Class Reference . . . . .	10
5.3.1	Detailed Description . . . . .	12
5.3.2	Member Typedef Documentation . . . . .	12
5.3.3	Constructor & Destructor Documentation . . . . .	13
5.3.4	Member Function Documentation . . . . .	14
5.4	rl_Traits::rl_DielectricPOD Struct Reference . . . . .	21
5.4.1	Detailed Description . . . . .	22
5.4.2	Member Data Documentation . . . . .	22

---

5.5	rl_DielectricPODArray Class Reference . . . . .	23
5.5.1	Detailed Description . . . . .	24
5.5.2	Constructor & Destructor Documentation . . . . .	24
5.5.3	Member Function Documentation . . . . .	25
5.5.4	Member Data Documentation . . . . .	27
5.6	rl_Exception Class Reference . . . . .	28
5.6.1	Detailed Description . . . . .	28
5.6.2	Constructor & Destructor Documentation . . . . .	28
5.7	rl_Multilayer Class Reference . . . . .	29
5.7.1	Detailed Description . . . . .	30
5.7.2	Member Typedef Documentation . . . . .	31
5.7.3	Constructor & Destructor Documentation . . . . .	31
5.7.4	Member Function Documentation . . . . .	31
5.7.5	Member Data Documentation . . . . .	34
5.8	rl_MultilayerSurface Class Reference . . . . .	35
5.8.1	Detailed Description . . . . .	37
5.8.2	Constructor & Destructor Documentation . . . . .	37
5.8.3	Member Function Documentation . . . . .	37
5.9	rl_Polarization Class Reference . . . . .	39
5.9.1	Detailed Description . . . . .	40
5.9.2	Member Typedef Documentation . . . . .	40
5.9.3	Constructor & Destructor Documentation . . . . .	41
5.9.4	Member Function Documentation . . . . .	42
5.10	rl_PolCSPOD Struct Reference . . . . .	45
5.10.1	Detailed Description . . . . .	46
5.10.2	Member Function Documentation . . . . .	46
5.10.3	Member Data Documentation . . . . .	48
5.11	rl_Ray Class Reference . . . . .	48
5.11.1	Detailed Description . . . . .	49
5.11.2	Constructor & Destructor Documentation . . . . .	50
5.11.3	Member Function Documentation . . . . .	50

5.12	rl_ReflectionCoefPOD Class Reference	54
5.12.1	Detailed Description	54
5.12.2	Member Function Documentation	54
5.13	rl_Traits Class Reference	57
5.13.1	Detailed Description	58
5.13.2	Member Typedef Documentation	58
5.13.3	Member Enumeration Documentation	58
5.14	rl_TransmissionCoefPOD Class Reference	59
5.14.1	Detailed Description	60
5.14.2	Member Function Documentation	60

## 1 rl\_RayLib User's Guide

### 1.1 Copyright and License

Copyright (C) 2006 Smithsonian Astrophysical Observatory

This file is part of the rl\_raylib package.

rl\_raylib is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

tracefct is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor Boston, MA 02110-1301, USA

#### Author

Terry Gaetz

### 1.2 Purpose

The rl\_RayLib library consists of a set of C++ classes for manipulating rays, including the effects of multilayer reflectivity.

### 1.2.1 Ray class

The [rl\\_Ray](#) class generalizes the [rl\\_BasicRay](#) class (found in the [rl\\_basicray](#) package) to include polarization information to the [rl\\_BasicRay](#)'s position, direction, energy, and id number components. See the [rl\\_basicray](#) package documentation for further information on the [rl\\_BasicRay](#) and [rl\\_RayMath](#) classes.

The rays can be translated/rotated from a standard coordinate system (STD) to a “body center system” (BCS), and de-rotated/de-translated from the BCS system back to the STD system. Given a surface normal, the ray direction can be reflected to a new direction. This yields much of the transformation functionality needed for basic raytracing.

### 1.2.2 Reflectivity classes

The reflectivity classes include a number of components:

- [rl\\_DielectricData](#) encapsulates an array of energy bins providing the dielectric decrement information and methods to evaluate (interpolate) the decrements at a requested energy. The array is built on a helper “Plain Ol’ Data” (POD) struct [rl\\_DielectricPOD](#) which provides the dielectric decrement at a specific energy.
- [rl\\_DielectricLayer](#) encapsulates the information about the interaction of a photon with a single dielectric layer, including the layer thickness, dielectric decrements given the photon energy, the component of the photon wave vector perpendicular to the layer, and various reflection and transmission coefficients, and surface “roughness” information. The layer can be a “substrate” (in which case the layer is considered as semi-infinite), vacuum, or a dielectric layer. These are mediated by helper “Plain Ol’ Data” (POD) structs and classes: [rl\\_ReflectionCoefPOD](#), [rl\\_TransmissionCoefPOD](#), [rl\\_ReflectionCoefPOD](#).
- [rl\\_Multilayer](#) encapsulates a stack of [rl\\_DielectricLayer](#)'s. Given the energy, sine of the graze angle, the multilayer reflectivity can be evaluated.
- [rl\\_MultilayerSurface](#) adds surface normal information to an [rl\\_Multilayer](#). Given an [rl\\_Ray](#), [rl\\_MultilayerSurface](#) can evaluate the reflectivity for the surface. This is also where hooks for surface interception and scattering could be placed.

## 2 Directory Hierarchy

### 2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

[rl\\_raylib](#)

4

## 3 Class Index

### 3.1 Class List

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

<a href="#">rl_DielectricData</a>	5
<a href="#">rl_DielectricData::rl_DielectricDataBinPOD</a>	9
<a href="#">rl_DielectricLayer</a>	10
<a href="#">rl_Traits::rl_DielectricPOD</a>	21
<a href="#">rl_DielectricPODArray</a>	23
<a href="#">rl_Exception</a>	28
<a href="#">rl_Multilayer</a>	29
<a href="#">rl_MultilayerSurface</a>	35
<a href="#">rl_Polarization</a>	39
<a href="#">rl_PolCSPOD</a>	45
<a href="#">rl_Ray</a>	48
<a href="#">rl_ReflectionCoefPOD</a>	54
<a href="#">rl_Traits</a>	57
<a href="#">rl_TransmissionCoefPOD</a>	59

## 4 Directory Documentation

### 4.1 rl\_raylib/ Directory Reference

Directory dependency graph for rl\_raylib/:



#### Files

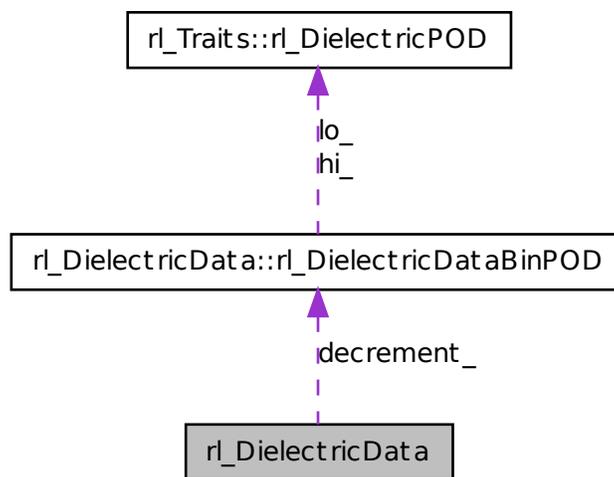
- file **rl\_DielectricData.cc**
- file **rl\_DielectricData.h**
- file **rl\_DielectricLayer.cc**
- file **rl\_DielectricLayer.h**
- file **rl\_DielectricPODArray.cc**
- file **rl\_DielectricPODArray.h**
- file **rl\_Exception.cc**
- file **rl\_Exception.h**
- file **rl\_Multilayer.cc**
- file **rl\_Multilayer.h**
- file **rl\_MultilayerSurface.cc**
- file **rl\_MultilayerSurface.h**
- file **rl\_Polarization.cc**
- file **rl\_Polarization.h**
- file **rl\_Ray.cc**
- file **rl\_Ray.h**
- file **rl\_RayLib.h**
- file **rl\_ReflectionCoefPOD.h**
- file **rl\_Traits.h**
- file **rl\_TransmissionCoefPOD.h**

## 5 Class Documentation

### 5.1 rl\_DielectricData Class Reference

```
#include <rl_DielectricData.h>
```

Collaboration diagram for rl\_DielectricData:



#### Classes

- struct [rl\\_DielectricDataBinPOD](#)

#### Public Member Functions

- [~rl\\_DielectricData\(\)](#)
- [rl\\_DielectricData\(\)](#)
- [rl\\_DielectricData\(rl\\_DielectricData const &other\) throw \(rl\\_Exception\)](#)
- [rl\\_DielectricData\(rl\\_Traits::rl\\_DielectricPOD const \\*diel, size\\_t num\\_pts, rl\\_Traits::EInterpMode interp\\_mode, double bulk\\_density=1.0\) throw \(rl\\_Exception\)](#)

- void `init` (`rl_Traits::rl_DielectricPOD` const \*diel, `size_t` num\_pts, `rl_Traits::EInterpMode` interp\_mode, double bulk\_density=1.0) throw ( `rl_Exception` )
- int `alpha_gamma` (double energy, double &alpha, double &gamma)
- double `energy_min` () const
- double `energy_max` () const
- double `bulk_density_factor` () const
- `rl_Traits::Bool` `is_vacuum` () const

### 5.1.1 Detailed Description

A class encapsulating the dielectric data (alpha, gamma) as a function of energy.

Definition at line 64 of file `rl_DielectricData.h`.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 `rl_DielectricData::~~rl_DielectricData` ( )

Non-virtual destructor.

Definition at line 53 of file `rl_DielectricData.cc`.

#### 5.1.2.2 `rl_DielectricData::rl_DielectricData` ( ) [`inline`]

Default constructor. Constructs a vacuum `rl_DielectricData`. Use `init` to initialize the object to a state other than vacuum.

Definition at line 215 of file `rl_DielectricData.h`.

#### 5.1.2.3 `rl_DielectricData::rl_DielectricData` ( `rl_DielectricData` const & *other* ) throw ( `rl_Exception` )

Copy constructor

### Parameters

*other* `rl_DielectricData` to be copied.

Definition at line 60 of file `rl_DielectricData.cc`.

#### 5.1.2.4 rl\_DielectricData::rl\_DielectricData ( rl\_Traits::rl\_DielectricPOD const \* *diel*, size\_t *num\_pts*, rl\_Traits::EInterpMode *interp\_mode*, double *bulk\_density* = 1.0 ) throw ( rl\_Exception )

Constructor. The data for the dielectric constants are obtained from a c-style array of const rl\_DielectricPOD; the array contains num\_pts points. The interpolation mode will be set to interp\_mode. The optical constants will be scaled by bulk\_density.

#### Parameters

- diel* an array of num\_pts POD's containing the complex dielectric decrements.
- num\_pts* number of elements in diel.
- interp\_mode* type of interpolation to be used in interpolating the dielectric constants.
- bulk\_density* relative bulk density factor to be used; 1.0 for full bulk density. The dielectric decrements will be scaled by bulk\_density.

Definition at line 75 of file rl\_DielectricData.cc.

### 5.1.3 Member Function Documentation

#### 5.1.3.1 int rl\_DielectricData::alpha\_gamma ( double *energy*, double & *alpha*, double & *gamma* )

Evaluate the dielectric decrements, alpha and gamma, at the given energy. The energy must be between the minimum and the maximum energies in the rl\_DielectricPOD array used to initialize the rl\_DielectricData Returns: 0 on success; -1 if energy is below lower limit, +1 if energy is above upper limit.

#### Parameters

- energy* photon energy (in keV).
- alpha* real part of the complex dielectric decrement.
- gamma* imaginary part of the complex dielectric decrement.

Definition at line 176 of file rl\_DielectricData.cc.

References rl\_Traits::rl\_DielectricPOD::alpha\_, rl\_Traits::ELinLin, rl\_Traits::ELinLog, rl\_Traits::ELogLin, rl\_Traits::ELogLog, rl\_Traits::rl\_DielectricPOD::energy\_, rl\_Traits::rl\_DielectricPOD::gamma\_, rl\_DielectricData::rl\_DielectricDataBinPOD::hi\_, and rl\_DielectricData::rl\_DielectricDataBinPOD::lo\_.

Referenced by rl\_DielectricLayer::setup\_for().

**5.1.3.2 double rl\_DielectricData::bulk\_density\_factor ( ) const [inline]**

Return the maximum energy covered by this dataset. Return the relative bulk density for this set of optical constants. 1.0 is nominal full bulk density

Definition at line 239 of file rl\_DielectricData.h.

Referenced by rl\_DielectricLayer::bulk\_density\_factor(), rl\_DielectricLayer::cdump\_on(), rl\_DielectricLayer::cprint\_constraints\_on(), and rl\_DielectricLayer::dump\_on().

**5.1.3.3 double rl\_DielectricData::energy\_max ( ) const [inline]**

Return the maximum energy covered by this dataset.

Definition at line 234 of file rl\_DielectricData.h.

Referenced by rl\_DielectricLayer::cdump\_on(), rl\_DielectricLayer::cprint\_constraints\_on(), rl\_DielectricLayer::dump\_on(), and rl\_DielectricLayer::energy\_max().

**5.1.3.4 double rl\_DielectricData::energy\_min ( ) const [inline]**

Return the minimum energy covered by this dataset.

Definition at line 229 of file rl\_DielectricData.h.

Referenced by rl\_DielectricLayer::cdump\_on(), rl\_DielectricLayer::cprint\_constraints\_on(), rl\_DielectricLayer::dump\_on(), and rl\_DielectricLayer::energy\_min().

**5.1.3.5 void rl\_DielectricData::init ( rl\_Traits::rl\_DielectricPOD const \* *diel*, size\_t *num\_pts*, rl\_Traits::EInterpMode *interp\_mode*, double *bulk\_density* = 1.0 ) throw ( rl\_Exception )**

Initialization function. The data are obtained from a c-style array of const rl\_DielectricPOD; the array contains num\_pts points. The interpolation mode will be set to interp\_mode. The optical constants will be scaled by bulk\_density.

**Parameters**

*diel* an array of num\_pts POD's containing the complex dielectric decrements.

*num\_pts* number of elements in diel.

*interp\_mode* type of interpolation to be used in interpolating the dielectric constants.

*bulk\_density* relative bulk density factor to be used; 1.0 for full bulk density. The dielectric decrements will be scaled by bulk\_density.

Definition at line 86 of file rl\_DielectricData.cc.

References rl\_Traits::ELinLin, rl\_Traits::ELinLog, rl\_Traits::ELogLin, and rl\_Traits::ELogLog.

### 5.1.3.6 rl\_Traits::Bool rl\_DielectricData::is\_vacuum ( ) const [inline]

Returns True if the object is a vacuum state; False, otherwise.

Definition at line 244 of file rl\_DielectricData.h.

Referenced by rl\_DielectricLayer::is\_vacuum().

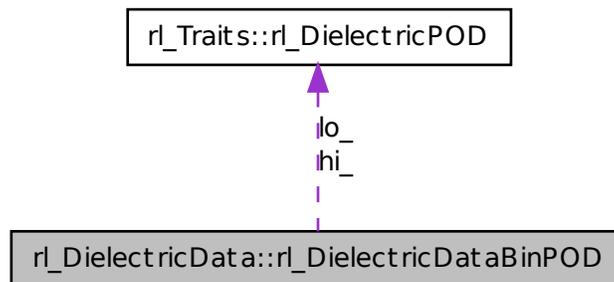
The documentation for this class was generated from the following files:

- rl\_DielectricData.h
- rl\_DielectricData.cc

## 5.2 rl\_DielectricData::rl\_DielectricDataBinPOD Struct Reference

```
#include <rl_DielectricData.h>
```

Collaboration diagram for rl\_DielectricData::rl\_DielectricDataBinPOD:



### Public Attributes

- [rl\\_Traits::rl\\_DielectricPOD lo\\_](#)  
*lower edge of energy bin*

- [rl\\_Traits::rl\\_DielectricPOD hi\\_](#)  
*upper edge of energy bin*

### 5.2.1 Detailed Description

A POD describing the lower and upper edge of an energy bin.

Definition at line 73 of file rl\_DielectricData.h.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 rl\_Traits::rl\_DielectricPOD rl\_DielectricData::rl\_DielectricDataBinPOD::hi\_

upper edge of energy bin

Definition at line 78 of file rl\_DielectricData.h.

Referenced by rl\_DielectricData::alpha\_gamma().

#### 5.2.2.2 rl\_Traits::rl\_DielectricPOD rl\_DielectricData::rl\_DielectricDataBinPOD::lo\_

lower edge of energy bin

Definition at line 76 of file rl\_DielectricData.h.

Referenced by rl\_DielectricData::alpha\_gamma().

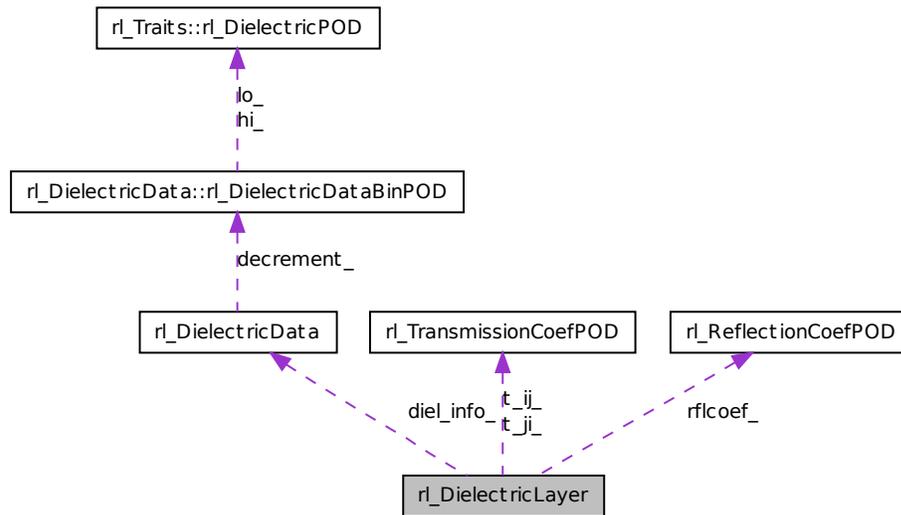
The documentation for this struct was generated from the following file:

- rl\_DielectricData.h

## 5.3 rl\_DielectricLayer Class Reference

```
#include <rl_DielectricLayer.h>
```

Collaboration diagram for rl\_DielectricLayer:



### Public Types

- typedef [rl\\_Traits::complex](#) `complex`  
*complex type*
- typedef [rl\\_Traits::ERoughType](#) `ERoughType`  
*roughness type*

### Public Member Functions

- [~rl\\_DielectricLayer](#) ()
- [rl\\_DielectricLayer](#) (char const layer\_name[ ]=0) throw ( [rl\\_Exception](#) )
- [rl\\_DielectricLayer](#) ( [rl\\_DielectricLayer](#) const &other) throw ( [rl\\_Exception](#) )
- [rl\\_DielectricLayer](#) ( [rl\\_Traits::rl\\_DielectricPOD](#) const \*diel, std::size\_t ndielpts, double layer\_thickness, double roughness, [rl\\_Traits::ERoughType](#) roughness\_type, [rl\\_Traits::EInterpMode](#) interp\_mode, double bulkdensity, char const \*layer\_name, [rl\\_Traits::Bool](#) is\_substrate=[rl\\_Traits::False](#)) throw ( [rl\\_Exception](#) )

- void `init` (rl\_Traits::rl\_DielectricPOD const \*diel, std::size\_t ndielpts, double layer\_thickness, double roughness, rl\_Traits::ERoughType roughness\_type, rl\_Traits::EInterpMode interp\_mode, double bulkdensity, char const \*layer\_name, rl\_Traits::Bool is\_substrate=rl\_Traits::False) throw ( rl\_Exception )
- int `setup_for` (double energy, double sinphi)
- void `reflect_nlayer` (rl\_DielectricLayer layer[], int num)
- void `reflect_amp` (rl\_DielectricLayer const &layer, double sinphi)
- complex const & `propagator` () const
- double `alpha` () const
- double `gamma` () const
- double `roughness` () const
- ERoughType `roughness_type` () const
- rl\_ReflectionCoefPOD const & `reflection_coef` () const
- double `reflectivity` (double polarization\_factor=0.0) const
- rl\_Traits::Bool `is_substrate` () const
- rl\_Traits::Bool `is_vacuum` () const
- char const \* `layer_name` () const
- double `energy_min` () const
- double `energy_max` () const
- double `thickness` () const
- double `zcoat` () const
- double `bulk_density_factor` () const
- std::ostream & `dump_on` (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const
- void `cdump_on` (std::FILE \*of, char const pre[ ]="", char const pst[ ]="") const
- void `cprint_constraints_on` (std::FILE \*of, char const pre[ ]="", char const pst[ ]="") const

### 5.3.1 Detailed Description

A class encapsulating the multilayer reflection of a ray. Given the energy and the sine of the graze angle, the reflection coefficient is evaluated.

Definition at line 60 of file rl\_DielectricLayer.h.

### 5.3.2 Member Typedef Documentation

#### 5.3.2.1 typedef rl\_Traits::complex rl\_DielectricLayer::complex

complex type

Definition at line 65 of file rl\_DielectricLayer.h.

### 5.3.2.2 typedef rl\_Traits::ERoughType rl\_DielectricLayer::ERoughType

roughness type

Definition at line 68 of file rl\_DielectricLayer.h.

### 5.3.3 Constructor & Destructor Documentation

#### 5.3.3.1 rl\_DielectricLayer::~~rl\_DielectricLayer ( )

Destructor

Definition at line 47 of file rl\_DielectricLayer.cc.

#### 5.3.3.2 rl\_DielectricLayer::rl\_DielectricLayer ( char const *layer\_name*[ ] = 0 ) throw ( rl\_Exception )

Constructor. By default, constructs a VACUUM layer. Use init method to initialize for conditions other than vacuum.

##### Parameters

*layer\_name* optional string naming the layer.

Definition at line 56 of file rl\_DielectricLayer.cc.

#### 5.3.3.3 rl\_DielectricLayer::rl\_DielectricLayer ( rl\_DielectricLayer const & *other* ) throw ( rl\_Exception )

DEEP Copy constructor

##### Parameters

*other* [rl\\_DielectricLayer](#) to be copied.

Definition at line 99 of file rl\_DielectricLayer.cc.

```

5.3.3.4 rl_DielectricLayer::rl_DielectricLayer ( rl_Traits::rl_DielectricPOD
const * diel, std::size_t ndielpts, double layer_thickness,
double roughness, rl_Traits::ERoughType roughness_type,
rl_Traits::EInterpMode interp_mode, double bulkdensity, char const *
layer_name, rl_Traits::Bool is_substrate = rl_Traits::False )
throw ( rl_Exception )

```

Constructor.

#### Parameters

*diel* array of dielectric decrement information.  
*ndielpts* number of points in diel array.  
*layer\_thickness* layer thickness (Angstroms).  
*roughness* layer “roughness” (Angstroms).  
*roughness\_type* interlayer grading “roughness” type.  
*interp\_mode* type of optical constant interpolation to be done.  
*bulkdensity* relative bulk density factor; 1 for full bulk density.  
*layer\_name* string describing the layer composition.  
*is\_substrate* boolean: True if this is the substrate layer.

Definition at line 142 of file rl\_DielectricLayer.cc.

#### 5.3.4 Member Function Documentation

```

5.3.4.1 double rl_DielectricLayer::alpha ( ) const [inline]

```

Returns this layer’s dielectric decrement (real part).

#### Returns

this layer’s dielectric decrement (real part).

Definition at line 528 of file rl\_DielectricLayer.h.

```

5.3.4.2 double rl_DielectricLayer::bulk_density_factor ( ) const [inline]

```

#### Returns

the relative bulk density for this layer. 1.0 is nominal full bulk density

Definition at line 580 of file rl\_DielectricLayer.h.

References rl\_DielectricData::bulk\_density\_factor().

**5.3.4.3** `void rl_DielectricLayer::cdump_on ( std::FILE * of, char const pre[] = "", char const pst[] = "" ) const`

Dumps layer information to a C-style FILE\* stream.

#### Parameters

- of* output file.
- pre* optional prefix string.
- pst* optional postfix string.

Definition at line 836 of file rl\_DielectricLayer.cc.

References rl\_DielectricData::bulk\_density\_factor(), rl\_ReflectionCoefPOD::cprint\_on(), rl\_DielectricData::energy\_max(), rl\_DielectricData::energy\_min(), and is\_vacuum().

Referenced by rl\_Multilayer::cdump\_on().

**5.3.4.4** `void rl_DielectricLayer::cprint_constraints_on ( std::FILE * of, char const pre[] = "", char const pst[] = "" ) const`

Dumps layer information and constraints to a C-style FILE\* stream.

#### Parameters

- of* output file.
- pre* optional prefix string.
- pst* optional postfix string.

Definition at line 880 of file rl\_DielectricLayer.cc.

References rl\_DielectricData::bulk\_density\_factor(), rl\_DielectricData::energy\_max(), rl\_DielectricData::energy\_min(), is\_vacuum(), rl\_ReflectionCoefPOD::para(), rl\_TransmissionCoefPOD::para(), rl\_ReflectionCoefPOD::perp(), and rl\_TransmissionCoefPOD::perp().

Referenced by reflect\_nlayer().

**5.3.4.5** `std::ostream & rl_DielectricLayer::dump_on ( std::ostream & os, char const pre[] = "", char const pst[] = "" ) const`

Dumps layer information to a stream.

#### Parameters

- os* stream.

*pre* optional prefix string.

*pst* optional postfix string.

Definition at line 791 of file `rl_DielectricLayer.cc`.

References `rl_DielectricData::bulk_density_factor()`, `rl_DielectricData::energy_max()`, `rl_DielectricData::energy_min()`, and `is_vacuum()`.

Referenced by `rl_Multilayer::dump_on()`.

#### 5.3.4.6 `double rl_DielectricLayer::energy_max ( ) const [inline]`

##### Returns

the maximum energy allowed for this layer.

Definition at line 576 of file `rl_DielectricLayer.h`.

References `rl_DielectricData::energy_max()`.

#### 5.3.4.7 `double rl_DielectricLayer::energy_min ( ) const [inline]`

##### Returns

the minimum energy allowed for this layer.

Definition at line 572 of file `rl_DielectricLayer.h`.

References `rl_DielectricData::energy_min()`.

#### 5.3.4.8 `double rl_DielectricLayer::gamma ( ) const [inline]`

Returns this layer's dielectric decrement (imag part).

##### Returns

this layer's dielectric decrement (imag part).

Definition at line 532 of file `rl_DielectricLayer.h`.

```

5.3.4.9 void rl_DielectricLayer::init ( rl_Traits::rl_DielectricPOD const *
    diel, std::size_t ndielpts, double layer_thickness, double roughness,
    rl_Traits::ERoughType roughness_type, rl_Traits::EInterpMode
    interp_mode, double bulkdensity, char const * layer_name,
    rl_Traits::Bool is_substrate = rl_Traits::False ) throw (
    rl_Exception )

```

Initializer

**Parameters**

*diel* array of dielectric decrement information.

*ndielpts* number of points in diel array.

*layer\_thickness* layer thickness (Angstroms).

*roughness* interlayer grading "roughness" (Angstrom).

*roughness\_type* interlayer grading "roughness" type.

*interp\_mode* type of optical constant interpolation to be done.

*bulkdensity* relative bulk density factor; 1 for full bulk density.

*layer\_name* string describing the layer composition.

*is\_substrate* boolean: True if this is the substrate layer.

Definition at line 160 of file rl\_DielectricLayer.cc.

References rl\_Traits::ERoughDebyeWaller\_CSAO, rl\_Traits::ERoughDebyeWaller\_RSAAO, rl\_Traits::ERoughDebyeWaller\_Spiller, rl\_Traits::ERoughModifiedDebyeWaller, rl\_Traits::ERoughNevotCroce, and rl\_Traits::ERoughNone.

```

5.3.4.10 rl_Traits::Bool rl_DielectricLayer::is_substrate ( ) const [inline]

```

**Returns**

rl\_Traits::True if this layer is the substrate, rl\_Traits::False otherwise.

Definition at line 560 of file rl\_DielectricLayer.h.

Referenced by reflect\_amp(), and setup\_for().

```

5.3.4.11 rl_Traits::Bool rl_DielectricLayer::is_vacuum ( ) const [inline]

```

**Returns**

rl\_Traits::True if this layer is the vacuum, rl\_Traits::False otherwise.

Definition at line 564 of file rl\_DielectricLayer.h.

References rl\_DielectricData::is\_vacuum().

Referenced by cdump\_on(), cprint\_constraints\_on(), dump\_on(), and setup\_for().

**5.3.4.12 char const \* rl\_DielectricLayer::layer\_name ( ) const [inline]****Returns**

the name of this layer as a character string.

Definition at line 568 of file rl\_DielectricLayer.h.

**5.3.4.13 rl\_Traits::complex const & rl\_DielectricLayer::propagator ( ) const [inline]**

The propagator for this layer.

**Returns**

propagator for this layer.

Definition at line 524 of file rl\_DielectricLayer.h.

Referenced by reflect\_nlayer().

**5.3.4.14 void rl\_DielectricLayer::reflect\_amp ( rl\_DielectricLayer const & layer, double sinphi )**

Compute reflection amplitude for the interface between this layer and the layer immediately above it. sinphi is the graze angle at the topmost layer (i.e., in vacuum). Layer "layer" is assumed to be adjacent to this layer, with "layer" nearer the vacuum and this layer nearer the substrate. The reflection coefficient for this layer is updated.

PRECONDITION: The layers must be pre-initialized with setup\_for

**Parameters**

*layer* adjacent layer above this layer, where above means closer to the vacuum.

*sinphi* sine of the graze angle between the ray (in vacuum) and the top surface of the multilayer.

Definition at line 324 of file rl\_DielectricLayer.cc.

References rl\_Traits::ERoughDebyeWaller\_CSAO, rl\_Traits::ERoughDebyeWaller\_RSAAO, rl\_Traits::ERoughDebyeWaller\_Spiller, rl\_Traits::ERoughModifiedDebyeWaller, rl\_Traits::ERoughNevotCroce, rl\_Traits::ERoughNone, is\_substrate(), rl\_TransmissionCoefPOD::para(), rl\_ReflectionCoefPOD::para(), rl\_TransmissionCoefPOD::perp(), and rl\_ReflectionCoefPOD::perp().

Referenced by rl\_Multilayer::multilayer\_reflect\_coef().

#### 5.3.4.15 void rl\_DielectricLayer::reflect\_nlayer ( rl\_DielectricLayer layer[], int num )

Compute reflectivity for a stack of num layers.

PRECONDITION: The layers must be pre-initialized with setup\_for and the reflection coefficients evaluated with reflect\_amp.

##### Parameters

*layer* array of layers.

*num* number of layers in the array.

Definition at line 520 of file rl\_DielectricLayer.cc.

References cprint\_constraints\_on(), rl\_ReflectionCoefPOD::para(), rl\_ReflectionCoefPOD::perp(), and propagator().

Referenced by rl\_Multilayer::multilayer\_reflect\_coef().

#### 5.3.4.16 rl\_ReflectionCoefPOD const & rl\_DielectricLayer::reflection\_coef ( ) const [inline]

Returns this layer's complex reflection coefficient.

##### Returns

this layer's complex reflection coefficient.

Definition at line 552 of file rl\_DielectricLayer.h.

Referenced by rl\_Multilayer::multilayer\_reflect\_coef().

**5.3.4.17** `double rl_DielectricLayer::reflectivity ( double polarization_factor = 0.0 ) const [inline]`

Returns: this layer's reflectivity.

#### Parameters

*polarization\_factor* - polarization factor; it must be a value between -1 and 1.

The polarization factor is related to parallel (p) and perpendicular (s) polarization by:  $\text{polarization\_factor} = (I_{\text{perp}} - I_{\text{para}}) / (I_{\text{perp}} + I_{\text{para}})$  or  $\text{polarization\_factor} = (I_s - I_p) / (I_s + I_p)$  where  $I_{\text{perp}}$  and  $I_{\text{para}}$  are the perpendicular and parallel E-field intensities, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

#### Returns

this layer's reflectivity

Definition at line 556 of file rl\_DielectricLayer.h.

References rl\_ReflectionCoefPOD::reflectivity().

**5.3.4.18** `double rl_DielectricLayer::roughness ( ) const [inline]`

Returns the roughness parameter of the upper surface of this layer.

#### Returns

the roughness parameter of the upper surface of this layer.

Definition at line 548 of file rl\_DielectricLayer.h.

**5.3.4.19** `rl_Traits::ERoughType rl_DielectricLayer::roughness_type ( ) const [inline]`

Returns the roughness type of the upper surface of this layer.

#### Returns

the roughness type of the upper surface of this layer.

Definition at line 544 of file rl\_DielectricLayer.h.

#### 5.3.4.20 int rl\_DielectricLayer::setup\_for ( double *energy*, double *sinphi* )

Set up layer state for given energy and graze angle.

##### Parameters

*energy* energy (keV).

*sinphi* sine of the graze angle between the ray (in vacuum) and the top surface of the multilayer.

Definition at line 224 of file rl\_DielectricLayer.cc.

References rl\_DielectricData::alpha\_gamma(), rl\_Traits::ERoughNone, is\_substrate(), and is\_vacuum().

Referenced by rl\_Multilayer::multilayer\_reflect\_coef().

#### 5.3.4.21 double rl\_DielectricLayer::thickness ( ) const [inline]

##### Returns

the layer thickness.

Definition at line 536 of file rl\_DielectricLayer.h.

#### 5.3.4.22 double rl\_DielectricLayer::zcoat ( ) const [inline]

##### Returns

the layer dimensionless thickness.

Definition at line 540 of file rl\_DielectricLayer.h.

The documentation for this class was generated from the following files:

- rl\_DielectricLayer.h
- rl\_DielectricLayer.cc

## 5.4 rl\_Traits::rl\_DielectricPOD Struct Reference

```
#include <rl_Traits.h>
```

### Public Attributes

- double [energy\\_](#)  
*energy (keV)*
- double [alpha\\_](#)  
*dielectric decrement, real part*
- double [gamma\\_](#)  
*dielectric decrement, imag part*

#### 5.4.1 Detailed Description

Plain Ol' Data: dielectric data (alpha, gamma) as a function of energy.

Definition at line 93 of file rl\_Traits.h.

#### 5.4.2 Member Data Documentation

##### 5.4.2.1 double rl\_Traits::rl\_DielectricPOD::alpha\_

dielectric decrement, real part

Definition at line 98 of file rl\_Traits.h.

Referenced by rl\_DielectricData::alpha\_gamma(), and rl\_DielectricPODArray::init().

##### 5.4.2.2 double rl\_Traits::rl\_DielectricPOD::energy\_

energy (keV)

Definition at line 96 of file rl\_Traits.h.

Referenced by rl\_DielectricData::alpha\_gamma(), and rl\_DielectricPODArray::init().

##### 5.4.2.3 double rl\_Traits::rl\_DielectricPOD::gamma\_

dielectric decrement, imag part

Definition at line 100 of file rl\_Traits.h.

Referenced by `rl_DielectricData::alpha_gamma()`, and `rl_DielectricPODArray::init()`.

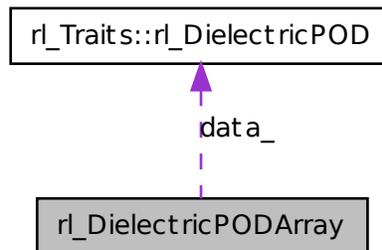
The documentation for this struct was generated from the following file:

- `rl_Traits.h`

## 5.5 rl\_DielectricPODArray Class Reference

```
#include <rl_DielectricPODArray.h>
```

Collaboration diagram for `rl_DielectricPODArray`:



### Public Member Functions

- `~rl_DielectricPODArray ()`
- `rl_DielectricPODArray ()`
- `rl_DielectricPODArray (size_t nelts, double const *energy, double const *alpha, double const *gamma) throw ( rl_Exception )`
- `rl_DielectricPODArray (size_t nelts, rl_Traits::rl_DielectricPOD *diel) throw ( rl_Exception )`
- `void init (size_t nelts, double const *energy, double const *alpha, double const *gamma)`
- `void init (size_t nelts, rl_Traits::rl_DielectricPOD *diel)`
- `size_t num_elts () const`
- `rl_Traits::rl_DielectricPOD const * const_data_ptr () const`
- `void cprint_on (std::FILE *of, char const pre[ ]="", char const pst[ ]="") const`

### Protected Attributes

- `size_t nelts_`  
*number of dielectric decrements read in*
- `rl_Traits::rl_DielectricPOD * data_`  
*pointer to the data*

#### 5.5.1 Detailed Description

A class encapsulating an array of `rl_DielectricPODs` storing the dielectric constants as a function of energy:

- energy (keV)
- alpha (real part of dielectric decrement)
- gamma (imaginary part of the dielectric decrement)

The complex dielectric constant has real part (1-alpha) and imaginary part (-gamma).

Definition at line 57 of file `rl_DielectricPODArray.h`.

#### 5.5.2 Constructor & Destructor Documentation

##### 5.5.2.1 `rl_DielectricPODArray::~~rl_DielectricPODArray ( )`

Destructor

Definition at line 65 of file `rl_DielectricPODArray.cc`.

References `data_`.

##### 5.5.2.2 `rl_DielectricPODArray::rl_DielectricPODArray ( )`

Default constructor.

An empty uninitialized `rl_DielectricPODArray` is created and the `init` method must be called to initialize the object.

Definition at line 69 of file `rl_DielectricPODArray.cc`.

**5.5.2.3** `rl_DielectricPODArray::rl_DielectricPODArray ( size_t nelts, double const * energy, double const * alpha, double const * gamma ) throw ( rl_Exception )`

Constructor.

#### Parameters

*nelts* number of elements in the array  
*energy* array of energies  
*alpha* array of dielectric decrement real part (alpha)  
*gamma* array of dielectric decrement imag part (gamma)

Definition at line 74 of file rl\_DielectricPODArray.cc.

**5.5.2.4** `rl_DielectricPODArray::rl_DielectricPODArray ( size_t nelts, rl_Traits::rl_DielectricPOD * diel ) throw ( rl_Exception )`

Constructor.

#### Parameters

*nelts* number of elements in the array  
*diel* array of dielectric decrement PODs

Definition at line 101 of file rl\_DielectricPODArray.cc.

### 5.5.3 Member Function Documentation

**5.5.3.1** `rl_Traits::rl_DielectricPOD const * rl_DielectricPODArray::const_data_ptr ( ) const [inline]`

Accessor.

#### Returns

pointer-to-const to rl\_DielectricPOD array.

Definition at line 174 of file rl\_DielectricPODArray.h.

References data\_.

**5.5.3.2** void rl\_DielectricPODArray::cprint\_on ( std::FILE \* *of*, char const *pre*[] = "", char const *pst*[] = "" ) const

Accessor.

#### Returns

pointer-to-const to rl\_DielectricPOD array. Print reflectivity information to output FILE\* stream.

#### Parameters

*of* output FILE\* stream.  
*pre* optional prefix (char\*) string.  
*pst* optional postfix (char\*) string.

Definition at line 156 of file rl\_DielectricPODArray.cc.

References data\_, and nelts\_.

**5.5.3.3** void rl\_DielectricPODArray::init ( size\_t *nelts*, rl\_Traits::rl\_DielectricPOD \* *diel* )

Initializer.

#### Parameters

*nelts* number of elements in the array  
*diel* array of dielectric decrement PODs

initialize this [rl\\_DielectricPODArray](#) from the input array of rl\_DielectricPODs. The [rl\\_DielectricPODArray](#) is sorted on the energy field.

Definition at line 140 of file rl\_DielectricPODArray.cc.

References rl\_Traits::rl\_DielectricPOD::alpha\_, data\_, rl\_Traits::rl\_DielectricPOD::energy\_, rl\_Traits::rl\_DielectricPOD::gamma\_, and nelts\_.

**5.5.3.4** void rl\_DielectricPODArray::init ( size\_t *nelts*, double const \* *energy*, double const \* *alpha*, double const \* *gamma* )

Initializer.

#### Parameters

*nelts* number of elements in the array  
*energy* array of energies

*alpha* array of dielectric decrement real part (alpha)

*gamma* array of dielectric decrement imag part (gamma)

initialize this [rl\\_DielectricPODArray](#) from the energy, alpha, and gamma arrays. The [rl\\_DielectricPODArray](#) is sorted on the energy field.

Definition at line 122 of file `rl_DielectricPODArray.cc`.

References `rl_Traits::rl_DielectricPOD::alpha_`, `data_`, `rl_Traits::rl_DielectricPOD::energy_`, `rl_Traits::rl_DielectricPOD::gamma_`, and `nelts_`.

#### 5.5.3.5 `size_t rl_DielectricPODArray::num_elts ( ) const [inline]`

Accessor.

#### Returns

number of elements in the `rl_DielectricPOD` array.

Definition at line 170 of file `rl_DielectricPODArray.h`.

References `nelts_`.

### 5.5.4 Member Data Documentation

#### 5.5.4.1 `rl_Traits::rl_DielectricPOD* rl_DielectricPODArray::data_ [protected]`

pointer to the data

Definition at line 64 of file `rl_DielectricPODArray.h`.

Referenced by `const_data_ptr()`, `cprint_on()`, `init()`, and `~rl_DielectricPODArray()`.

#### 5.5.4.2 `size_t rl_DielectricPODArray::nelts_ [protected]`

number of dielectric decrements read in

Definition at line 62 of file `rl_DielectricPODArray.h`.

Referenced by `cprint_on()`, `init()`, and `num_elts()`.

The documentation for this class was generated from the following files:

- `rl_DielectricPODArray.h`
- `rl_DielectricPODArray.cc`

## 5.6 rl\_Exception Class Reference

```
#include <rl_Exception.h>
```

### Public Member Functions

- [~rl\\_Exception](#) () throw ()  
*Destructor.*
- [rl\\_Exception](#) (const string &msg)  
*Include a string describing the exception.*
- [rl\\_Exception](#) (const char \*format,...)  
*Format a string describing the exception.*

### 5.6.1 Detailed Description

The exception thrown by the rl\_RayLib and rl\_RaySupLib libraries.

Definition at line 35 of file rl\_Exception.h.

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 rl\_Exception::~~rl\_Exception ( ) throw ( )

Destructor.

Definition at line 27 of file rl\_Exception.cc.

#### 5.6.2.2 rl\_Exception::rl\_Exception ( const string & msg )

Include a string describing the exception.

Definition at line 29 of file rl\_Exception.cc.

#### 5.6.2.3 rl\_Exception::rl\_Exception ( const char \* format, ... )

Format a string describing the exception.

Definition at line 32 of file rl\_Exception.cc.

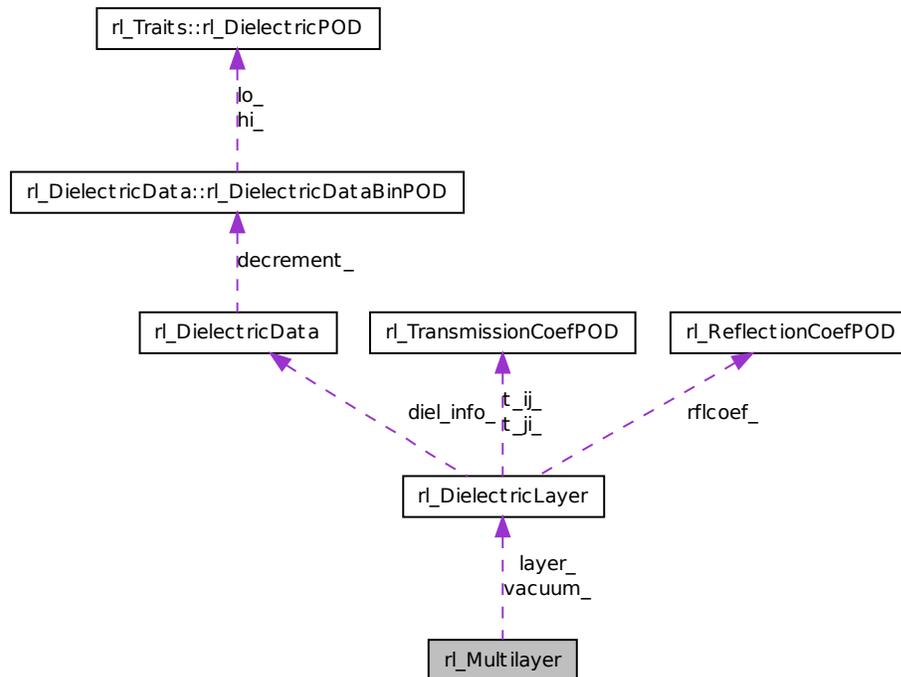
The documentation for this class was generated from the following files:

- rl\_Exception.h
- rl\_Exception.cc

## 5.7 rl\_Multilayer Class Reference

```
#include <rl_Multilayer.h>
```

Collaboration diagram for rl\_Multilayer:



### Public Types

- typedef [rl\\_Traits::EInterpMode](#) [EInterpMode](#)  
*interpolation mode for dielectric data*

**Public Member Functions**

- virtual `~rl_Multilayer()`
- `rl_Multilayer` (int num\_layers, `rl_DielectricLayer` \*layers, `rl_Traits::Bool` adopt\_data=`rl_Traits::True`)
- void `init` (int num\_layers, `rl_DielectricLayer` \*layers, `rl_Traits::Bool` adopt\_data=`rl_Traits::True`)
- int `multilayer_reflect_coef` (`rl_ReflectionCoefPOD` &rfl, double energy, double sg)
- int `multilayer_reflectivity` (double &rfl, double energy, double sg, double polarization\_factor=0.0)
- `rl_DielectricLayer` const & `layer` (int layer\_no) const
- int `num_layers` () const
- `std::ostream` & `dump_on` (`std::ostream` &os, int layer\_no, char const pre[ ]="", char const pst[ ]="") const
- void `cdump_on` (`std::FILE` \*of, int layer\_no, char const pre[ ]="", char const pst[ ]="") const

**Protected Attributes**

- int `num_layers_`  
*number of multilayers*
- `rl_DielectricLayer` `vacuum_`  
*the top (vacuum) layer*
- `rl_DielectricLayer` \* `layer_`  
*array of dielectric layers*
- `rl_Traits::Bool` `own_data_`  
*do we own the `rl_DielectricLayer` array?*

**5.7.1 Detailed Description**

A class encapsulating reflection of a ray from a multilayer surface. Given the energy and the sine of the graze angle, reflection coefficient can be evaluated.

Definition at line 60 of file `rl_Multilayer.h`.

## 5.7.2 Member Typedef Documentation

### 5.7.2.1 typedef rl\_Traits::EInterpMode rl\_Multilayer::EInterpMode

interpolation mode for dielectric data

Definition at line 85 of file rl\_Multilayer.h.

## 5.7.3 Constructor & Destructor Documentation

### 5.7.3.1 rl\_Multilayer::~~rl\_Multilayer ( ) [virtual]

Destructor.

Frees up [rl\\_DielectricLayer](#) array if own\_data\_ is rl\_Traits::True.

Definition at line 46 of file rl\_Multilayer.cc.

References [layer\\_](#), and [own\\_data\\_](#).

### 5.7.3.2 rl\_Multilayer::rl\_Multilayer ( int num\_layers, rl\_DielectricLayer \* layers, rl\_Traits::Bool adopt\_data = rl\_Traits::True )

Construct multilayer from num\_layers individual layers.

#### Parameters

*num\_layers* number of layers to construct.

*layers* pointer to array of layer data

*adopt\_data* boolean flag

- if rl\_Traits::True, [rl\\_Multilayer](#) assumes responsibility for deleting the [rl\\_DielectricLayer](#) array;
- if rl\_Traits::False, it is the caller's responsibility.

Definition at line 53 of file rl\_Multilayer.cc.

## 5.7.4 Member Function Documentation

### 5.7.4.1 void rl\_Multilayer::cdump\_on ( std::FILE \* of, int layer\_no, char const pre[] = "", char const pst[] = "" ) const

Dump information about layer layer\_no to output FILE\*.

**Parameters**

*of* output FILE\*  
*layer\_no* layer number to be dumped  
*pre* optional prefix string  
*pst* optional postfix string

Definition at line 181 of file rl\_Multilayer.cc.

References rl\_DielectricLayer::cdump\_on(), and layer\_.

#### 5.7.4.2 `std::ostream & rl_Multilayer::dump_on ( std::ostream & os, int layer_no, char const pre[] = "", char const pst[] = "" ) const`

Dump information about layer layer\_no to stream os.

**Returns**

reference to stream os.

**Parameters**

*os* output stream  
*layer\_no* layer number to be dumped  
*pre* optional prefix string  
*pst* optional postfix string

Definition at line 164 of file rl\_Multilayer.cc.

References rl\_DielectricLayer::dump\_on(), and layer\_.

Referenced by rl\_MultilayerSurface::dump\_on().

#### 5.7.4.3 `void rl_Multilayer::init ( int num_layers, rl_DielectricLayer * layers, rl_Traits::Bool adopt_data = rl_Traits::True )`

Initialize multilayer from num\_layers individual layers.

**Parameters**

*num\_layers* number of layers to construct.  
*layers* pointer to array of layer data  
*adopt\_data* boolean flag

- if rl\_Traits::True, [rl\\_Multilayer](#) assumes responsibility for deleting the [rl\\_DielectricLayer](#) array;

- if rl\_Traits::False, it is the caller's responsibility.

Definition at line 64 of file rl\_Multilayer.cc.

References layer\_, num\_layers\_, and own\_data\_.

#### 5.7.4.4 rl\_DielectricLayer const & rl\_Multilayer::layer ( int layer\_no ) const

##### Returns

const reference to layer layer\_no

Definition at line 156 of file rl\_Multilayer.cc.

References layer\_.

Referenced by rl\_MultilayerSurface::layer().

#### 5.7.4.5 int rl\_Multilayer::multilayer\_reflect\_coef ( rl\_ReflectionCoeffPOD & rfl, double energy, double sg )

Evaluate the multilayer reflection coefficients

##### Returns

0 if successful, the number of the invalid layer if not.

##### Parameters

*rfl* the computed reflection coefficient.

*energy* ray energy.

*sg* sine of the graze angle between the ray and the surface.

Definition at line 90 of file rl\_Multilayer.cc.

References layer\_, num\_layers\_, rl\_DielectricLayer::reflect\_amp(), rl\_DielectricLayer::reflect\_nlayer(), rl\_DielectricLayer::reflection\_coef(), rl\_DielectricLayer::setup\_for(), and vacuum\_.

Referenced by multilayer\_reflectivity(), and rl\_MultilayerSurface::reflect().

#### 5.7.4.6 int rl\_Multilayer::multilayer\_reflectivity ( double & rfl, double energy, double sg, double polarization\_factor = 0.0 )

Evaluate the multilayer reflectivity (assuming unpolarized rays)

**Returns**

0 if successful, the number of the invalid layer if not.

**Parameters**

*rfl* multilayer reflectivity.

*energy* ray energy.

*sg* sine of the graze angle between the ray and the surface.

*polarization\_factor* polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:

$$\text{polarization\_factor} = \frac{(I_{\perp} - I_{\parallel})}{(I_{\perp} + I_{\parallel})}$$

or

$$\text{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$$

where  $I_{\perp}$  and  $I_{\parallel}$  are the perpendicular and parallel E-field *intensities*, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

Definition at line 136 of file rl\_Multilayer.cc.

References rl\_ReflectionCoefPOD::init(), multilayer\_reflect\_coef(), and rl\_ReflectionCoefPOD::reflectivity().

**5.7.4.7 int rl\_Multilayer::num\_layers ( ) const [inline]****Returns**

number of layers

Definition at line 231 of file rl\_Multilayer.h.

References num\_layers\_.

**5.7.5 Member Data Documentation****5.7.5.1 rl\_DielectricLayer\* rl\_Multilayer::layer\_ [protected]**

array of dielectric layers

Definition at line 78 of file rl\_Multilayer.h.

Referenced by `cdump_on()`, `dump_on()`, `init()`, `layer()`, `multilayer_reflect_coef()`, and `~rl_Multilayer()`.

#### 5.7.5.2 int rl\_Multilayer::num\_layers\_ [protected]

number of multilayers

Definition at line 74 of file rl\_Multilayer.h.

Referenced by `init()`, `multilayer_reflect_coef()`, and `num_layers()`.

#### 5.7.5.3 rl\_Traits::Bool rl\_Multilayer::own\_data\_ [protected]

do we own the [rl\\_DielectricLayer](#) array?

Definition at line 80 of file rl\_Multilayer.h.

Referenced by `init()`, and `~rl_Multilayer()`.

#### 5.7.5.4 rl\_DielectricLayer rl\_Multilayer::vacuum\_ [protected]

the top (vacuum) layer

Definition at line 76 of file rl\_Multilayer.h.

Referenced by `multilayer_reflect_coef()`.

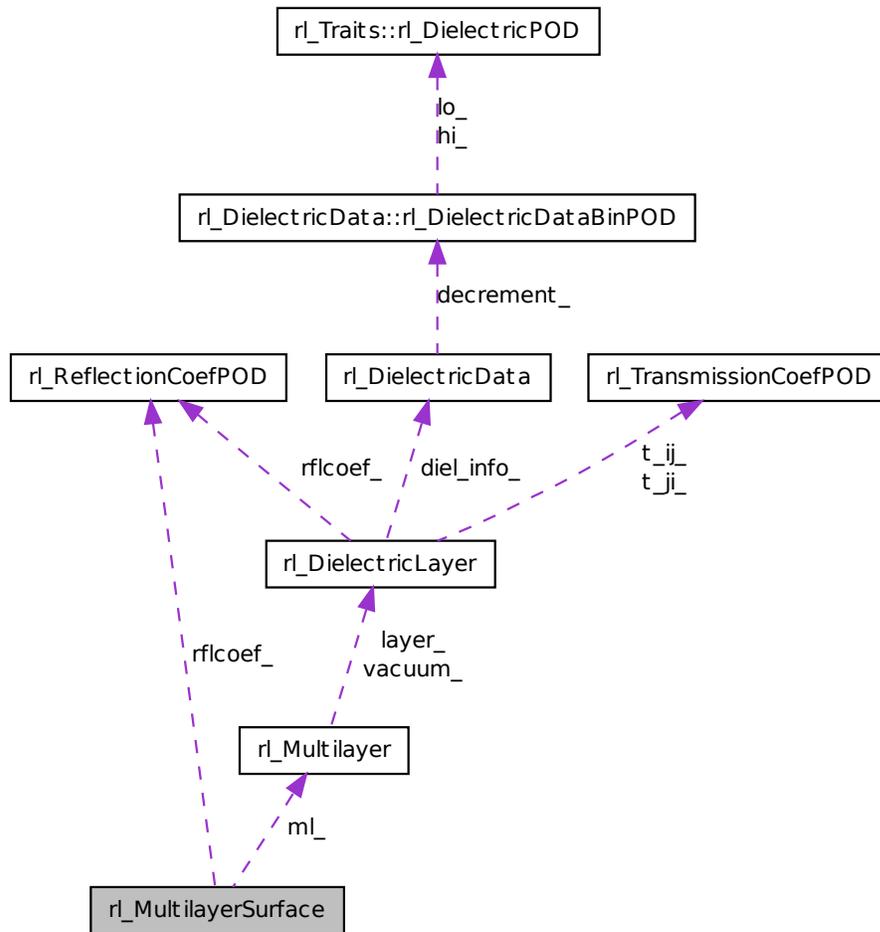
The documentation for this class was generated from the following files:

- rl\_Multilayer.h
- rl\_Multilayer.cc

## 5.8 rl\_MultilayerSurface Class Reference

```
#include <rl_MultilayerSurface.h>
```

Collaboration diagram for rl\_MultilayerSurface:



### Public Member Functions

- virtual `~rl_MultilayerSurface()`
- `rl_MultilayerSurface(rl_Multilayer &ml, dvm3_Vector &norm)`
- `dvm3_Vector const &normal_vector()` const
- void `set_normal(dvm3_Vector const &norm)`

- int `reflect` (rl\_Ray &ray)
- rl\_ReflectionCoefPOD const & `reflection_coefs` () const
- rl\_DielectricLayer const & `layer` (int layer\_no) const
- std::ostream & `dump_on` (std::ostream &os, int layer\_no, char const pre[]="", char const pst[]="") const

### 5.8.1 Detailed Description

A class encapsulating a multilayer surface, including surface normal. This class contains the multilayer data and surface normal *by reference*. That is, the `rl_Multilayer` and `dvm3_Vector` objects must already exist.

Definition at line 51 of file `rl_MultilayerSurface.h`.

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 rl\_MultilayerSurface::~rl\_MultilayerSurface ( ) [virtual]

Destructor.

Definition at line 35 of file `rl_MultilayerSurface.cc`.

#### 5.8.2.2 rl\_MultilayerSurface::rl\_MultilayerSurface ( rl\_Multilayer & ml, dvm3\_Vector & norm ) [inline]

Constructor. Constructs multilayer surface information from the provided `rl_Multilayer` object and normal vector.

#### Parameters

- ml* the `rl_Multilayer` object to be used (must already exist)
- norm* the surface normal `dvm3_Vector` to be used (must already exist)

Definition at line 181 of file `rl_MultilayerSurface.h`.

### 5.8.3 Member Function Documentation

#### 5.8.3.1 std::ostream & rl\_MultilayerSurface::dump\_on ( std::ostream & os, int layer\_no, char const pre[] = "", char const pst[] = "" ) const [inline]

Dump information about a layer.

### Parameters

- os* output stream
- layer\_no* layer number for which the information is requested.
- pre* optional char\* prefix string.
- pst* optional char\* postfix string.

Definition at line 231 of file rl\_MultilayerSurface.h.

References rl\_Multilayer::dump\_on().

#### 5.8.3.2 rl\_DielectricLayer const & rl\_MultilayerSurface::layer ( int layer\_no ) const [inline]

Accessor.

### Parameters

- layer\_no* layer number for which the information is requested.

### Returns

const reference to the [rl\\_DielectricLayer](#) for layer layer\_no.

Definition at line 226 of file rl\_MultilayerSurface.h.

References rl\_Multilayer::layer().

#### 5.8.3.3 dvm3\_Vector const & rl\_MultilayerSurface::normal\_vector ( ) const [inline]

Accessor.

### Returns

the surface normal vector at the ray intercept

Definition at line 190 of file rl\_MultilayerSurface.h.

#### 5.8.3.4 int rl\_MultilayerSurface::reflect ( rl\_Ray & ray ) [inline]

Mutator. Reflect ray direction vector at this surface

### Parameters

- ray* to be reflected. The ray is assumed to be already at a valid surface intercept point. The ray direction vector and polarization state are updated.

**Returns**

0 if successful.

Definition at line 213 of file rl\_MultilayerSurface.h.

References rl\_Multilayer::multilayer\_reflect\_coef(), and rl\_Ray::reflect().

### 5.8.3.5 rl\_ReflectionCoefPOD const & rl\_MultilayerSurface::reflection\_coefs ( ) const `[inline]`

Accessor.

**Returns**

[rl\\_ReflectionCoefPOD](#) holding the complex perpendicular and parallel reflection coefficients.

Definition at line 195 of file rl\_MultilayerSurface.h.

### 5.8.3.6 void rl\_MultilayerSurface::set\_normal ( dvm3\_Vector const & norm ) `[inline]`

Mutator. Set the surface normal at the ray/surface intercept. Until the full surface intercept apparatus is available, applications need to set the surface normal at the intercept.

**Parameters**

*norm* the surface normal vector at the ray intercept

Definition at line 203 of file rl\_MultilayerSurface.h.

The documentation for this class was generated from the following files:

- rl\_MultilayerSurface.h
- rl\_MultilayerSurface.cc

## 5.9 rl\_Polarization Class Reference

```
#include <rl_Polarization.h>
```

**Public Types**

- typedef [rl\\_Traits::complex](#) `complex`  
*complex type*

## Public Member Functions

- [~rl\\_Polarization](#) ()
- [rl\\_Polarization](#) ()
- [rl\\_Polarization](#) ([rl\\_Polarization](#) const &rhs)
- [rl\\_Polarization](#) ([rl\\_PolCSPOD](#) const &cs, [dvm3\\_Vector](#) const &dir)
- void [init](#) ([rl\\_PolCSPOD](#) const &cs, [dvm3\\_Vector](#) const &dir)
- void [get\\_PolCSPOD](#) ([rl\\_PolCSPOD](#) &cs, [dvm3\\_Vector](#) const &dir) const
- double [intensity](#) () const
- void [rotate](#) ([rl\\_Polarization](#) &rotated, [dvm3\\_RotMat](#) const &rot\_mtx)
- void [derotate](#) ([rl\\_Polarization](#) &derotated, [dvm3\\_RotMat](#) const &rot\_mtx)
- void [attenuate](#) (double factor)
- void [reflect](#) ([dvm3\\_Vector](#) const &normal, [dvm3\\_Vector](#) const &dir\_in, [dvm3\\_Vector](#) const &dir\_out, [rl\\_ReflectionCoefPOD](#) const &rflcoef)
- [dvm3\\_Vector](#) const & [C\\_r](#) () const  
*return the real "cosine" polarization vector*
- [dvm3\\_Vector](#) const & [C\\_i](#) () const  
*return the imaginary "cosine" polarization vector*
- [dvm3\\_Vector](#) const & [S\\_r](#) () const  
*return the real "sine" polarization vector*
- [dvm3\\_Vector](#) const & [S\\_i](#) () const  
*return the imaginary "sine" polarization vector*
- [std::ostream](#) & [print\\_on](#) ([std::ostream](#) &os) const
- void [cprint\\_on](#) ([std::FILE](#) \*of) const

### 5.9.1 Detailed Description

Encapsulates the polarization state of a ray. The class is a simple class to handle the ray polarization state resolved onto a specific coordinate system. The polarization axes are constructed from a direction vector and the coordinate axes. A coordinate triple is constructed in which one axis is the direction vector.

Definition at line 186 of file [rl\\_Polarization.h](#).

### 5.9.2 Member Typedef Documentation

#### 5.9.2.1 typedef [rl\\_Traits::complex](#) [rl\\_Polarization::complex](#)

complex type

Definition at line 191 of file `rl_Polarization.h`.

### 5.9.3 Constructor & Destructor Documentation

#### 5.9.3.1 `rl_Polarization::~~rl_Polarization ( ) [inline]`

Destructor (NON-VIRTUAL)

Definition at line 343 of file `rl_Polarization.h`.

#### 5.9.3.2 `rl_Polarization::rl_Polarization ( ) [inline]`

Default constructor; constructs `rl_Polarization` with INVALID fields.

Definition at line 349 of file `rl_Polarization.h`.

#### 5.9.3.3 `rl_Polarization::rl_Polarization ( rl_Polarization const & rhs ) [inline]`

Copy constructor; constructs `rl_Polarization` with INVALID fields.

#### Parameters

*rhs* - `rl_Polarization` object to be copied.

Definition at line 359 of file `rl_Polarization.h`.

#### 5.9.3.4 `rl_Polarization::rl_Polarization ( rl_PolCSPOD const & cs, dvm3_Vector const & dir ) [inline]`

Conversion constructor. Convert OSAC-style polarization amplitudes into polarization vector set.

#### Parameters

*cs* OSAC-style polarization amplitude

*dir* ray direction vector

Definition at line 354 of file `rl_Polarization.h`.

References `init()`.

## 5.9.4 Member Function Documentation

### 5.9.4.1 void rl\_Polarization::attenuate ( double *factor* )

Attenuate reflectivity by a factor.

#### Parameters

*factor* attenuation factor. (NOTE: this multiplies the polarization components by sqrt(factor).)

Definition at line 243 of file rl\_Polarization.cc.

Referenced by rl\_Ray::attenuate().

### 5.9.4.2 dvm3\_Vector const & rl\_Polarization::C\_i ( ) const [inline]

return the imaginary “cosine” polarization vector

Definition at line 395 of file rl\_Polarization.h.

### 5.9.4.3 dvm3\_Vector const & rl\_Polarization::C\_r ( ) const [inline]

return the real “cosine” polarization vector

Definition at line 388 of file rl\_Polarization.h.

### 5.9.4.4 void rl\_Polarization::cprint\_on ( std::FILE \* *of* ) const

Write the polarization amplitude to FILE\* of. The state is written as "[x y z][x y z][x y z][x y z]".

#### Parameters

*of* output FILE\*.

### 5.9.4.5 void rl\_Polarization::derotate ( rl\_Polarization & *derotated*, dvm3\_RotMat const & *rot\_mtx* ) [inline]

Rotate polarization state back from frame described by rot\_mtx

**Parameters**

*derotated* output derotated polarization state. The state is now back in the original frame.

*rot\_mtx* rotation matrix.

Definition at line 378 of file rl\_Polarization.h.

Referenced by rl\_Ray::derotate\_detranslate().

**5.9.4.6 void rl\_Polarization::get\_PolCSPOD ( rl\_PolCSPOD & cs, dvm3\_Vector const & dir ) const**

Evaluate [rl\\_PolCSPOD](#) corresponding to this polarization state.

**Parameters**

*cs* OSAC-style polarization amplitude

*dir* ray direction vector

Definition at line 217 of file rl\_Polarization.cc.

References [rl\\_PolCSPOD::c2\\_](#), and [rl\\_PolCSPOD::s2\\_](#).

Referenced by [rl\\_Ray::get\\_PolCSPOD\(\)](#).

**5.9.4.7 void rl\_Polarization::init ( rl\_PolCSPOD const & cs, dvm3\_Vector const & dir )**

Initialize the polarization state.

**Parameters**

*cs* OSAC-style polarization amplitude

*dir* ray direction vector

Definition at line 171 of file rl\_Polarization.cc.

References [rl\\_PolCSPOD::c2\\_](#), and [rl\\_PolCSPOD::s2\\_](#).

Referenced by [rl\\_Ray::init\\_ray\(\)](#), [rl\\_Polarization\(\)](#), and [rl\\_Ray::set\\_polarization\(\)](#).

**5.9.4.8 double rl\_Polarization::intensity ( ) const**

Evaluate intensity.

**Returns**

intensity (i.e., Stoke's  $s_0$ ).

Definition at line 208 of file rl\_Polarization.cc.

Referenced by rl\_Ray::intensity().

**5.9.4.9 std::ostream & rl\_Polarization::print\_on ( std::ostream & os ) const**

Write the polarization vectors to stream os. The state is written as "[x y z][x y z][x y z][x y z]".

**Parameters**

*os* output stream.

Definition at line 339 of file rl\_Polarization.cc.

Referenced by rl\_Ray::print\_on().

**5.9.4.10 void rl\_Polarization::reflect ( dvm3\_Vector const & normal, dvm3\_Vector const & dir\_in, dvm3\_Vector const & dir\_out, rl\_ReflectionCoefPOD const & rflcoef )**

Evaluate the reflected polarization vectors. Given a surface normal, incident and reflected ray direction vectors, and the parallel and perpendicular reflection coefficients, evaluate the reflected polarization vectors.

**Parameters**

*normal* surface normal vector.

*dir\_in* direction vector for incoming ray.

*dir\_out* direction vector for reflected ray.

*rflcoef* complex reflection coefficients for surface.

Definition at line 262 of file rl\_Polarization.cc.

References rl\_ReflectionCoefPOD::para(), and rl\_ReflectionCoefPOD::perp().

Referenced by rl\_Ray::reflect().

**5.9.4.11 void rl\_Polarization::rotate ( rl\_Polarization & rotated, dvm3\_RotMat const & rot\_mtx ) [inline]**

Rotate polarization state to frame described by rot\_mtx

**Parameters**

*rotated* output rotated polarization state. The state is now in the frame obtained by applying *rot\_mtx*.

*rot\_mtx* rotation matrix.

Definition at line 368 of file rl\_Polarization.h.

Referenced by rl\_Ray::translate\_rotate().

**5.9.4.12 dvm3\_Vector const & rl\_Polarization::S\_i ( ) const [inline]**

return the imaginary “sine” polarization vector

Definition at line 409 of file rl\_Polarization.h.

**5.9.4.13 dvm3\_Vector const & rl\_Polarization::S\_r ( ) const [inline]**

return the real “sine” polarization vector

Definition at line 402 of file rl\_Polarization.h.

The documentation for this class was generated from the following files:

- rl\_Polarization.h
- rl\_Polarization.cc

**5.10 rl\_PolCSPOD Struct Reference**

```
#include <rl_Polarization.h>
```

**Public Member Functions**

- void [init](#) (double b\_over\_a=1.0, double psi=0.0)
- void [attenuate](#) (double factor)
- double [intensity](#) () const
- std::ostream & [print\\_on](#) (std::ostream &os) const
- void [cprint\\_on](#) (FILE \*of) const

**Public Attributes**

- [rl\\_Traits::complex c2\\_ \[2\]](#)  
*polarization amplitude (cosine component)*
- [rl\\_Traits::complex s2\\_ \[2\]](#)  
*polarization amplitude (sine component)*

**5.10.1 Detailed Description**

A Plain Old Data struct (POD) encapsulating the OSAC-style complex polarization amplitudes. This is a utility POD to simplify communication with BPipe.

Relation to OSAC polarization amplitudes:

$$\begin{matrix} c2\_q1 & \iff & c2comp(1), & c2\_q2 & \iff & c2comp(2) & s2\_q1 & \iff & s2comp(1), & s2\_q2 & \iff & s2comp(2) \end{matrix}$$

For random polarization:

$$wt = c2.q1.r^2 + c2.q2.r^2 + c2.q1.i^2 + c2.q2.i^2 + s2.q1.r^2 + s2.q2.r^2 + s2.q1.i^2 + s2.q2.i^2$$

or

$$wt = |c2[0]|^2 + |s2[0]|^2$$

For discrete polarization:

$$wt = c2.q1.r^2 + c2.q2.r^2 + c2.q1.i^2 + c2.q2.i^2 + s2.q1.r^2 + s2.q2.r^2 + s2.q1.i^2 + s2.q2.i^2 + 2(c2.q1.i s2.q1.r + c2.q2.i s2.q2.r)$$

or

$$wt = |c2[0]|^2 + |s2[0]|^2 + 2(c2[0]s2*[0] + c2[1]s2*[1])$$

where the \* superscript denotes a complex conjugate.

Definition at line 106 of file rl\_Polarization.h.

**5.10.2 Member Function Documentation****5.10.2.1 void rl\_PolCSPOD::attenuate ( double *factor* )**

Attenuate intensity by a factor.

**Parameters**

*factor* the attenuation factor; the ray intensity is multiplied by factor.

Definition at line 121 of file rl\_Polarization.cc.

References [c2\\_](#), and [s2\\_](#).

### 5.10.2.2 void rl\_PolCSPOD::cprint\_on ( FILE \* *of* ) const

Write the polarization amplitude to FILE\* of. The state is written out as "[r,i)(r,i)]\n[r,i)(r,i)]".

#### Parameters

*of* output FILE\*.

Definition at line 149 of file rl\_Polarization.cc.

References c2\_, intensity(), and s2\_.

### 5.10.2.3 void rl\_PolCSPOD::init ( double *b\_over\_a* = 1.0, double *psi* = 0.0 )

Initialization method.

#### Parameters

*b\_over\_a* ratio of semiminor axis to semi-major axis.

*psi* angle of the major axis from the +X axis, with psi increasing towards +Y

Definition at line 76 of file rl\_Polarization.cc.

References c2\_, and s2\_.

### 5.10.2.4 double rl\_PolCSPOD::intensity ( ) const

Evaluate the intensity.

#### Returns

the intensity (i.e., Stoke's  $s_0$ ).

Definition at line 99 of file rl\_Polarization.cc.

References c2\_, and s2\_.

Referenced by cprint\_on().

### 5.10.2.5 std::ostream & rl\_PolCSPOD::print\_on ( std::ostream & *os* ) const

Write the polarization amplitude to stream os. The state is written out as "[r,i)(r,i)][r,i)(r,i)]".

#### Parameters

*os* output stream.

Definition at line 135 of file rl\_Polarization.cc.

References `c2_`, and `s2_`.

### 5.10.3 Member Data Documentation

#### 5.10.3.1 rl\_Traits::complex rl\_PolCSPOD::c2\_[2]

polarization amplitude (cosine component)

Definition at line 109 of file rl\_Polarization.h.

Referenced by `attenuate()`, `cprint_on()`, `rl_Polarization::get_PolCSPOD()`, `rl_Polarization::init()`, `init()`, `intensity()`, and `print_on()`.

#### 5.10.3.2 rl\_Traits::complex rl\_PolCSPOD::s2\_[2]

polarization amplitude (sine component)

Definition at line 111 of file rl\_Polarization.h.

Referenced by `attenuate()`, `cprint_on()`, `rl_Polarization::get_PolCSPOD()`, `rl_Polarization::init()`, `init()`, `intensity()`, and `print_on()`.

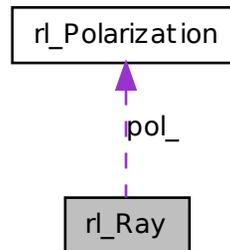
The documentation for this struct was generated from the following files:

- `rl_Polarization.h`
- `rl_Polarization.cc`

## 5.11 rl\_Ray Class Reference

```
#include <rl_Ray.h>
```

Collaboration diagram for rl\_Ray:



### Public Member Functions

- virtual [~rl\\_Ray](#) ()
- [rl\\_Ray](#) ()
- [rl\\_Ray](#) (rl\_BasicRay const &ray, [rl\\_PolCSPOD](#) const &cspol)
- [rl\\_Ray](#) (dvm3\_Vector const &pos, dvm3\_Vector const &dir, double energy, long int id, [rl\\_PolCSPOD](#) const &cspol)
- void [init\\_ray](#) (dvm3\_Vector const &pos, dvm3\_Vector const &dir, double energy, long int id, [rl\\_PolCSPOD](#) const &cspol)
- [rl\\_Polarization](#) const & [polarization](#) () const
- void [set\\_polarization](#) ([rl\\_PolCSPOD](#) const &cspol)
- void [get\\_PolCSPOD](#) ([rl\\_PolCSPOD](#) &cs) const
- double [intensity](#) () const
- void [attenuate](#) (double by\_how\_much)
- void [reflect](#) (dvm3\_Vector const &normal, [rl\\_ReflectionCoefPOD](#) const &rflcoef)
- void [translate\\_rotate](#) (dvm3\_Vector const &trans, dvm3\_RotMat const &rotmat)
- void [derotate\\_detranslate](#) (dvm3\_Vector const &trans, dvm3\_RotMat const &rotmat)
- std::ostream & [print\\_on](#) (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const

#### 5.11.1 Detailed Description

An [rl\\_BasicRay](#) with added polarization information.

Definition at line 45 of file [rl\\_Ray.h](#).

## 5.11.2 Constructor & Destructor Documentation

### 5.11.2.1 rl\_Ray::~~rl\_Ray ( ) [virtual]

Destructor

Definition at line 41 of file rl\_Ray.cc.

### 5.11.2.2 rl\_Ray::rl\_Ray ( ) [inline]

Default constructor. Constructs a ray in an INVALID state; use init\_ray to initialize the fields.

Definition at line 193 of file rl\_Ray.h.

### 5.11.2.3 rl\_Ray::rl\_Ray ( rl\_BasicRay const & ray, rl\_PolCSPOD const & cspol ) [inline]

Constructor. Construct a ray given an rl\_BasicRay and a polarization state.

Definition at line 198 of file rl\_Ray.h.

### 5.11.2.4 rl\_Ray::rl\_Ray ( dvm3\_Vector const & pos, dvm3\_Vector const & dir, double energy, long int id, rl\_PolCSPOD const & cspol ) [inline]

Constructor. Construct a ray.

#### Parameters

*pos* ray position vector.

*dir* ray direction unit vector.

*energy* ray energy (keV).

*id* a numeric ray identifier.

*cspol* an OSAC-style complex polarization amplitude vector.

Definition at line 203 of file rl\_Ray.h.

## 5.11.3 Member Function Documentation

### 5.11.3.1 void rl\_Ray::attenuate ( double by\_how\_much ) [inline]

Attenuate this ray by by\_how\_much

**Parameters**

*by\_how\_much* how much to attenuate this ray.

Definition at line 247 of file rl\_Ray.h.

References rl\_Polarization::attenuate().

### 5.11.3.2 void rl\_Ray::derotate\_detranslate ( dvm3\_Vector const & *trans*, dvm3\_RotMat const & *rotmat* )

Derotate back to std coordinates; detranslate back to std origin

**Parameters**

*trans* translation vector.

*rotmat* rotation matrix to be applied. *rotmat* specifies a rotation from std to bes; the inverse of *rotmat* is applied to the ray.

Definition at line 75 of file rl\_Ray.cc.

References rl\_Polarization::derotate().

### 5.11.3.3 void rl\_Ray::get\_PolCSPOD ( rl\_PolCSPOD & *cs* ) const [inline]

Evaluate [rl\\_PolCSPOD](#) corresponding to this ray's polarization state.

**Parameters**

*cs* return an OSAC-style complex polarization amplitude vector.

Definition at line 229 of file rl\_Ray.h.

References rl\_Polarization::get\_PolCSPOD().

### 5.11.3.4 void rl\_Ray::init\_ray ( dvm3\_Vector const & *pos*, dvm3\_Vector const & *dir*, double *energy*, long int *id*, rl\_PolCSPOD const & *cspol* ) [inline]

Initialize a ray.

**Parameters**

*pos* ray position vector.

*dir* ray direction unit vector.

*energy* ray energy (keV).  
*id* a numeric ray identifier.  
*cspol* an OSAC-style complex polarization amplitude vector.

Definition at line 211 of file `rl_Ray.h`.

References `rl_Polarization::init()`.

#### 5.11.3.5 `double rl_Ray::intensity ( ) const [inline]`

Returns this ray's normalized intensity (i.e., "weight")

##### Returns

this ray's normalized intensity (i.e., "weight")

Definition at line 234 of file `rl_Ray.h`.

References `rl_Polarization::intensity()`.

#### 5.11.3.6 `rl_Polarization const & rl_Ray::polarization ( ) const [inline]`

Return the polarization state.

##### Returns

polarization state

Definition at line 224 of file `rl_Ray.h`.

#### 5.11.3.7 `std::ostream & rl_Ray::print_on ( std::ostream & os, char const pre[] = "", char const pst[] = "" ) const`

Write the ray contents with optional pre and post comment strings.

##### Parameters

*os* output stream.  
*pre* optional prefix c-string.  
*pst* optional postfix c-string.

Definition at line 86 of file `rl_Ray.cc`.

References `rl_Polarization::print_on()`.

**5.11.3.8 void rl\_Ray::reflect ( dvm3\_Vector const & *normal*,  
rl\_ReflectionCoefPOD const & *rlcoef* )**

Reflect this ray's direction vector and polarization state.

**Parameters**

*normal* surface normal unit vector.

*rlcoef* complex reflectances for the surface.

Definition at line 51 of file rl\_Ray.cc.

References rl\_Polarization::reflect().

Referenced by rl\_MultilayerSurface::reflect().

**5.11.3.9 void rl\_Ray::set\_polarization ( rl\_PolCSPOD const & *cspol* )  
[inline]**

Set the polarization state.

**Parameters**

*cspol* an OSAC-style complex polarization amplitude vector.

Definition at line 242 of file rl\_Ray.h.

References rl\_Polarization::init().

**5.11.3.10 void rl\_Ray::translate\_rotate ( dvm3\_Vector const & *trans*,  
dvm3\_RotMat const & *rotmat* )**

Translate to BCS origin and rotate from STD to BCS coordinates.

**Parameters**

*trans* translation vector

*rotmat* rotation matrix to be applied. rotmat specifies a rotation from std to bcs.

Definition at line 63 of file rl\_Ray.cc.

References rl\_Polarization::rotate().

The documentation for this class was generated from the following files:

- rl\_Ray.h
- rl\_Ray.cc

## 5.12 rl\_ReflectionCoefPOD Class Reference

```
#include <rl_ReflectionCoefPOD.h>
```

### Public Member Functions

- double [reflectivity](#) (double polarization\_factor=0.0) const
- void [init](#) ()
- void [init](#) (rl\_Traits::complex &para, rl\_Traits::complex &perp)
- [rl\\_Traits::complex](#) para () const
- [rl\\_Traits::complex](#) & para ()
- [rl\\_Traits::complex](#) perp () const
- [rl\\_Traits::complex](#) & perp ()
- std::ostream & [print\\_on](#) (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const
- void [cprint\\_on](#) (std::FILE \*of, char const pre[ ]="", char const pst[ ]="") const

### 5.12.1 Detailed Description

A Plain Old C++ Data class representing complex reflection coefficients.

Definition at line 76 of file rl\_ReflectionCoefPOD.h.

### 5.12.2 Member Function Documentation

**5.12.2.1 void rl\_ReflectionCoefPOD::cprint\_on ( std::FILE \* of, char const pre[ ] = "", char const pst[ ] = "" ) const [inline]**

Print reflectivity information to output FILE\* stream.

#### Parameters

- of* output FILE\* stream.
- pre* optional prefix (char\*) string.
- pst* optional postfix (char\*) string.

Definition at line 219 of file rl\_ReflectionCoefPOD.h.

References [para\(\)](#), and [perp\(\)](#).

Referenced by [rl\\_DielectricLayer::cdump\\_on\(\)](#).

**5.12.2.2 void rl\_ReflectionCoefPOD::init ( ) [inline]**

initialize perpendicular (s) and parallel (p) reflection coefficients to zero.

Definition at line 172 of file rl\_ReflectionCoefPOD.h.

Referenced by rl\_Multilayer::multilayer\_reflectivity().

**5.12.2.3 void rl\_ReflectionCoefPOD::init ( rl\_Traits::complex & para, rl\_Traits::complex & perp ) [inline]**

initialize perpendicular (s) and parallel (p) reflection coefficients. to zero.

**Parameters**

*para* parallel reflection coefficient.

*perp* perpendicular reflection coefficient.

Definition at line 179 of file rl\_ReflectionCoefPOD.h.

**5.12.2.4 rl\_Traits::complex rl\_ReflectionCoefPOD::para ( ) const [inline]****Returns**

parallel (p) reflection coefficient.

Definition at line 187 of file rl\_ReflectionCoefPOD.h.

Referenced by rl\_DielectricLayer::cprint\_constraints\_on(), cprint\_on(), rl\_Polarization::reflect(), rl\_DielectricLayer::reflect\_amp(), and rl\_DielectricLayer::reflect\_nlayer().

**5.12.2.5 rl\_Traits::complex & rl\_ReflectionCoefPOD::para ( ) [inline]****Returns**

parallel (p) reflection coefficient (read/write access).

Definition at line 195 of file rl\_ReflectionCoefPOD.h.

### 5.12.2.6 rl\_Traits::complex & rl\_ReflectionCoefPOD::perp ( ) [inline]

#### Returns

perpendicular (p) reflection coefficient (read/write access).

Definition at line 199 of file rl\_ReflectionCoefPOD.h.

### 5.12.2.7 rl\_Traits::complex rl\_ReflectionCoefPOD::perp ( ) const [inline]

#### Returns

perpendicular (p) reflection coefficient.

Definition at line 191 of file rl\_ReflectionCoefPOD.h.

Referenced by rl\_DielectricLayer::cprint\_constraints\_on(), cprint\_on(), rl\_Polarization::reflect(), rl\_DielectricLayer::reflect\_amp(), and rl\_DielectricLayer::reflect\_nlayer().

### 5.12.2.8 std::ostream & rl\_ReflectionCoefPOD::print\_on ( std::ostream & os, char const pre[] = "", char const pst[] = "" ) const [inline]

Print reflectivity information to output stream.

#### Parameters

*os* output stream.

*pre* optional prefix (char\*) string.

*pst* optional postfix (char\*) string.

Definition at line 210 of file rl\_ReflectionCoefPOD.h.

### 5.12.2.9 double rl\_ReflectionCoefPOD::reflectivity ( double polarization\_factor = 0.0 ) const [inline]

evaluate the reflectivity.

**Parameters**

*polarization\_factor* polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:

$$\text{polarization\_factor} = \frac{(I_{\perp} - I_{\parallel})}{(I_{\perp} + I_{\parallel})}$$

or

$$\text{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$$

where  $I_{\perp}$  and  $I_{\parallel}$  are the perpendicular and parallel E-field *intensities*, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

**Returns**

const reference to layer layer\_no

Definition at line 203 of file rl\_ReflectionCoefPOD.h.

Referenced by rl\_Multilayer::multilayer\_reflectivity(), and rl\_DielectricLayer::reflectivity().

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

- rl\_ReflectionCoefPOD.h

**5.13 rl\_Traits Class Reference**

```
#include <rl_Traits.h>
```

**Classes**

- struct [rl\\_DielectricPOD](#)

**Public Types**

- enum [Bool](#) { **False**, **True** }  
*Typedef for the Boolean type.*
- enum [EInterpMode](#) { [ELinLin](#), [ELinLog](#), [ELogLin](#), [ELogLog](#) }

- enum [ERoughType](#) {  
    [ERoughNone](#), [ERoughDebyeWaller\\_RSAO](#), [ERoughDebyeWaller\\_CSAO](#),  
    [ERoughDebyeWaller\\_Spiller](#),  
    [ERoughModifiedDebyeWaller](#), [ERoughNevotCroce](#) }
- typedef std::complex< double > [complex](#)  
    *Typedef for the complex type.*

### 5.13.1 Detailed Description

[rl\\_Traits](#) is a “traits” class for the [rl\\_RayLib](#) library. It defines typedefs (e.g., abstracting out the complex class) and enums used in the library. It also declares a Plain Old Data (POD) struct to encapsulate the dielectric constant data.

Definition at line 55 of file [rl\\_Traits.h](#).

### 5.13.2 Member Typedef Documentation

#### 5.13.2.1 typedef std::complex<double> rl\_Traits::complex

Typedef for the complex type.

Definition at line 61 of file [rl\\_Traits.h](#).

### 5.13.3 Member Enumeration Documentation

#### 5.13.3.1 enum rl\_Traits::Bool

Typedef for the Boolean type.

Definition at line 64 of file [rl\\_Traits.h](#).

#### 5.13.3.2 enum rl\_Traits::EInterpMode

Enumeration specifying the interpolation of the optical constants.

#### Enumerator:

*ELinLin* linear in energy, linear in optical constants.

*ELinLog* log in energy, linear in optical constants.

*ELogLin* linear in energy, log in optical constants.

*ELogLog* log in energy, log in optical constants.

Definition at line 69 of file rl\_Traits.h.

### 5.13.3.3 enum rl\_Traits::ERoughType

Enumeration specifying the type of interlayer diffusion treatment

#### Enumerator:

*ERoughNone* no interlayer diffusion

*ERoughDebyeWaller\_RSAO* Debye-Waller factor

*ERoughDebyeWaller\_CSAO* Debye-Waller factor

*ERoughDebyeWaller\_Spiller* Debye-Waller factor

*ERoughModifiedDebyeWaller* Modified Debye-Waller factor

*ERoughNevotCroce* Nevot-Croce factor

Definition at line 80 of file rl\_Traits.h.

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

- rl\_Traits.h

## 5.14 rl\_TransmissionCoefPOD Class Reference

```
#include <rl_TransmissionCoefPOD.h>
```

### Public Member Functions

- void [init](#) ()
- void [init](#) (rl\_Traits::complex &para, rl\_Traits::complex &perp)
- double [transmission](#) (double polarization\_factor=0.0) const
- rl\_Traits::complex [para](#) () const
- rl\_Traits::complex & [para](#) ()
- rl\_Traits::complex [perp](#) () const
- rl\_Traits::complex & [perp](#) ()
- std::ostream & [print\\_on](#) (std::ostream &os, char const pre[ ]="", char const pst[ ]="") const
- void [cprint\\_on](#) (std::FILE \*of, char const pre[ ]="", char const pst[ ]="") const

### 5.14.1 Detailed Description

A Plain Old Data class representing complex reflection coefficients.

Definition at line 75 of file rl\_TransmissionCoefPOD.h.

### 5.14.2 Member Function Documentation

**5.14.2.1** `void rl_TransmissionCoefPOD::cprint_on ( std::FILE * of, char const pre[] = "", char const pst[] = "" ) const [inline]`

Print reflectivity information to output FILE\* stream.

#### Parameters

- of* output FILE\* stream.
- pre* optional prefix (char\*) string.
- pst* optional postfix (char\*) string.

Definition at line 218 of file rl\_TransmissionCoefPOD.h.

**5.14.2.2** `void rl_TransmissionCoefPOD::init ( ) [inline]`

Initialize perpendicular (s) and parallel (p) transmission coefficients to zero.

Definition at line 171 of file rl\_TransmissionCoefPOD.h.

**5.14.2.3** `void rl_TransmissionCoefPOD::init ( rl_Traits::complex & para, rl_Traits::complex & perp ) [inline]`

Initialize perpendicular (s) and parallel (p) transmission coefficients.

#### Parameters

- para* parallel transmission coefficient
- perp* perpendicular transmission coefficient

Definition at line 178 of file rl\_TransmissionCoefPOD.h.

**5.14.2.4** `rl_Traits::complex rl_TransmissionCoefPOD::para ( ) const [inline]`

**Returns**

parallel (p) transmission coefficient.

Definition at line 186 of file rl\_TransmissionCoefPOD.h.

Referenced by rl\_DielectricLayer::cprint\_constraints\_on(), and rl\_DielectricLayer::reflect\_amp().

**5.14.2.5 rl\_Traits::complex & rl\_TransmissionCoefPOD::para ( )  
[inline]****Returns**

parallel (p) transmission coefficient (read/write access).

Definition at line 194 of file rl\_TransmissionCoefPOD.h.

**5.14.2.6 rl\_Traits::complex & rl\_TransmissionCoefPOD::perp ( )  
[inline]****Returns**

perpendicular (s) transmission coefficient (read/write access).

Definition at line 198 of file rl\_TransmissionCoefPOD.h.

**5.14.2.7 rl\_Traits::complex rl\_TransmissionCoefPOD::perp ( ) const  
[inline]****Returns**

perpendicular (s) transmission coefficient.

Definition at line 190 of file rl\_TransmissionCoefPOD.h.

Referenced by rl\_DielectricLayer::cprint\_constraints\_on(), and rl\_DielectricLayer::reflect\_amp().

**5.14.2.8** `std::ostream & rl_TransmissionCoefPOD::print_on ( std::ostream & os, char const pre[] = "", char const pst[] = "" ) const [inline]`

Print reflectivity information to output stream.

#### Parameters

- os* output stream.
- pre* optional prefix (char\*) string.
- pst* optional postfix (char\*) string.

Definition at line 209 of file rl\_TransmissionCoefPOD.h.

**5.14.2.9** `double rl_TransmissionCoefPOD::transmission ( double polarization_factor = 0.0 ) const [inline]`

Transmission factor.

#### Parameters

- polarization\_factor* polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:

$$\text{polarization\_factor} = \frac{(I_{\perp} - I_{\parallel})}{(I_{\perp} + I_{\parallel})}$$

or

$$\text{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$$

where  $I_{\perp}$  and  $I_{\parallel}$  are the perpendicular and parallel E-field *intensities*, respectively. Thus,

- -1: pure parallel (p) polarization.
- 0: completely unpolarized.
- +1: pure perpendicular (s) polarization.

#### Returns

transmission factor

Definition at line 202 of file rl\_TransmissionCoefPOD.h.

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

- rl\_TransmissionCoefPOD.h

## Index

- ~rl\_DielectricData
  - rl\_DielectricData, 6
- ~rl\_DielectricLayer
  - rl\_DielectricLayer, 13
- ~rl\_DielectricPODArray
  - rl\_DielectricPODArray, 24
- ~rl\_Exception
  - rl\_Exception, 28
- ~rl\_Multilayer
  - rl\_Multilayer, 31
- ~rl\_MultilayerSurface
  - rl\_MultilayerSurface, 37
- ~rl\_Polarization
  - rl\_Polarization, 41
- ~rl\_Ray
  - rl\_Ray, 50
- alpha
  - rl\_DielectricLayer, 14
- alpha\_
  - rl\_Traits::rl\_DielectricPOD, 22
- alpha\_gamma
  - rl\_DielectricData, 7
- attenuate
  - rl\_Polarization, 42
  - rl\_PolCSPOD, 46
  - rl\_Ray, 50
- Bool
  - rl\_Traits, 58
- bulk\_density\_factor
  - rl\_DielectricData, 7
  - rl\_DielectricLayer, 14
- c2\_
  - rl\_PolCSPOD, 48
- C\_i
  - rl\_Polarization, 42
- C\_r
  - rl\_Polarization, 42
- cdump\_on
  - rl\_DielectricLayer, 14
  - rl\_Multilayer, 31
- complex
  - rl\_DielectricLayer, 12
  - rl\_Polarization, 40
  - rl\_Traits, 58
- const\_data\_ptr
  - rl\_DielectricPODArray, 25
- cprint\_constraints\_on
  - rl\_DielectricLayer, 15
- cprint\_on
  - rl\_DielectricPODArray, 25
  - rl\_Polarization, 42
  - rl\_PolCSPOD, 46
  - rl\_ReflectionCoefPOD, 54
  - rl\_TransmissionCoefPOD, 60
- data\_
  - rl\_DielectricPODArray, 27
- derotate
  - rl\_Polarization, 42
- derotate\_detranslate
  - rl\_Ray, 51
- dump\_on
  - rl\_DielectricLayer, 15
  - rl\_Multilayer, 32
  - rl\_MultilayerSurface, 37
- EInterpMode
  - rl\_Multilayer, 31
  - rl\_Traits, 58
- ELinLin
  - rl\_Traits, 58
- ELinLog
  - rl\_Traits, 58
- ELogLin
  - rl\_Traits, 58
- ELogLog
  - rl\_Traits, 59
- energy\_
  - rl\_Traits::rl\_DielectricPOD, 22
- energy\_max
  - rl\_DielectricData, 7
  - rl\_DielectricLayer, 16
- energy\_min

- rl\_DielectricData, 8
  - rl\_DielectricLayer, 16
- ERoughDebyeWaller\_CSAO
  - rl\_Traits, 59
- ERoughDebyeWaller\_RSAO
  - rl\_Traits, 59
- ERoughDebyeWaller\_Spiller
  - rl\_Traits, 59
- ERoughModifiedDebyeWaller
  - rl\_Traits, 59
- ERoughNevotCroce
  - rl\_Traits, 59
- ERoughNone
  - rl\_Traits, 59
- ERoughType
  - rl\_DielectricLayer, 12
  - rl\_Traits, 59
- gamma
  - rl\_DielectricLayer, 16
- gamma\_
  - rl\_Traits::rl\_DielectricPOD, 22
- get\_PolCSPOD
  - rl\_Polarization, 43
  - rl\_Ray, 51
- hi\_
  - rl\_DielectricData::rl\_DielectricDataBinPOD, 10
- init
  - rl\_DielectricData, 8
  - rl\_DielectricLayer, 16
  - rl\_DielectricPODArray, 26
  - rl\_Multilayer, 32
  - rl\_Polarization, 43
  - rl\_PolCSPOD, 47
  - rl\_ReflectionCoefPOD, 54, 55
  - rl\_TransmissionCoefPOD, 60
- init\_ray
  - rl\_Ray, 51
- intensity
  - rl\_Polarization, 43
  - rl\_PolCSPOD, 47
  - rl\_Ray, 52
- is\_substrate
  - rl\_DielectricLayer, 17
- is\_vacuum
  - rl\_DielectricData, 8
  - rl\_DielectricLayer, 17
- layer
  - rl\_Multilayer, 33
  - rl\_MultilayerSurface, 38
- layer\_
  - rl\_Multilayer, 34
- layer\_name
  - rl\_DielectricLayer, 18
- lo\_
  - rl\_DielectricData::rl\_DielectricDataBinPOD, 10
- multilayer\_reflect\_coef
  - rl\_Multilayer, 33
- multilayer\_reflectivity
  - rl\_Multilayer, 33
- nelts\_
  - rl\_DielectricPODArray, 27
- normal\_vector
  - rl\_MultilayerSurface, 38
- num\_elts
  - rl\_DielectricPODArray, 27
- num\_layers
  - rl\_Multilayer, 34
- num\_layers\_
  - rl\_Multilayer, 35
- own\_data\_
  - rl\_Multilayer, 35
- para
  - rl\_ReflectionCoefPOD, 55
  - rl\_TransmissionCoefPOD, 60, 61
- perp
  - rl\_ReflectionCoefPOD, 55, 56
  - rl\_TransmissionCoefPOD, 61
- polarization
  - rl\_Ray, 52
- print\_on
  - rl\_Polarization, 44
  - rl\_PolCSPOD, 47
  - rl\_Ray, 52

- rl\_ReflectionCoefPOD, 56
  - rl\_TransmissionCoefPOD, 61
- propagator
  - rl\_DielectricLayer, 18
- reflect
  - rl\_MultilayerSurface, 38
  - rl\_Polarization, 44
  - rl\_Ray, 52
- reflect\_amp
  - rl\_DielectricLayer, 18
- reflect\_nlayer
  - rl\_DielectricLayer, 19
- reflection\_coef
  - rl\_DielectricLayer, 19
- reflection\_coefs
  - rl\_MultilayerSurface, 39
- reflectivity
  - rl\_DielectricLayer, 19
  - rl\_ReflectionCoefPOD, 56
- rl\_Traits
  - ELinLin, 58
  - ELinLog, 58
  - ELogLin, 58
  - ELogLog, 59
  - ERoughDebyeWaller\_CSAO, 59
  - ERoughDebyeWaller\_RSAO, 59
  - ERoughDebyeWaller\_Spiller, 59
  - ERoughModifiedDebyeWaller, 59
  - ERoughNevotCroce, 59
  - ERoughNone, 59
- rl\_DielectricData, 4
  - ~rl\_DielectricData, 6
  - alpha\_gamma, 7
  - bulk\_density\_factor, 7
  - energy\_max, 7
  - energy\_min, 8
  - init, 8
  - is\_vacuum, 8
  - rl\_DielectricData, 6
  - rl\_DielectricData, 6
- rl\_DielectricData::rl\_-
  - DielectricDataBinPOD, 9
  - hi\_, 10
  - lo\_, 10
- rl\_DielectricLayer, 10
  - ~rl\_DielectricLayer, 13
  - alpha, 14
  - bulk\_density\_factor, 14
  - cdump\_on, 14
  - complex, 12
  - cprint\_constraints\_on, 15
  - dump\_on, 15
  - energy\_max, 16
  - energy\_min, 16
  - ERoughType, 12
  - gamma, 16
  - init, 16
  - is\_substrate, 17
  - is\_vacuum, 17
  - layer\_name, 18
  - propagator, 18
  - reflect\_amp, 18
  - reflect\_nlayer, 19
  - reflection\_coef, 19
  - reflectivity, 19
  - rl\_DielectricLayer, 13
  - rl\_DielectricLayer, 13
  - roughness, 20
  - roughness\_type, 20
  - setup\_for, 20
  - thickness, 21
  - zcoat, 21
- rl\_DielectricPODArray, 23
  - ~rl\_DielectricPODArray, 24
  - const\_data\_ptr, 25
  - cprint\_on, 25
  - data\_, 27
  - init, 26
  - nelts\_, 27
  - num\_elts, 27
  - rl\_DielectricPODArray, 24, 25
  - rl\_DielectricPODArray, 24, 25
- rl\_Exception, 28
  - ~rl\_Exception, 28
  - rl\_Exception, 28
  - rl\_Exception, 28
- rl\_Multilayer, 29
  - ~rl\_Multilayer, 31
  - cdump\_on, 31
  - dump\_on, 32
  - EInterpMode, 31

- init, 32
- layer, 33
- layer\_, 34
- multilayer\_reflect\_coef, 33
- multilayer\_reflectivity, 33
- num\_layers, 34
- num\_layers\_, 35
- own\_data\_, 35
- rl\_Multilayer, 31
- rl\_Multilayer, 31
- vacuum\_, 35
- rl\_MultilayerSurface, 35
  - ~rl\_MultilayerSurface, 37
  - dump\_on, 37
  - layer, 38
  - normal\_vector, 38
  - reflect, 38
  - reflection\_coefs, 39
  - rl\_MultilayerSurface, 37
  - rl\_MultilayerSurface, 37
  - set\_normal, 39
- rl\_Polarization, 39
  - ~rl\_Polarization, 41
  - attenuate, 42
  - C\_i, 42
  - C\_r, 42
  - complex, 40
  - cprint\_on, 42
  - derotate, 42
  - get\_PolCSPOD, 43
  - init, 43
  - intensity, 43
  - print\_on, 44
  - reflect, 44
  - rl\_Polarization, 41
  - rl\_Polarization, 41
  - rotate, 44
  - S\_i, 45
  - S\_r, 45
- rl\_PolCSPOD, 45
  - attenuate, 46
  - c2\_, 48
  - cprint\_on, 46
  - init, 47
  - intensity, 47
  - print\_on, 47
  - s2\_, 48
- rl\_Ray, 48
  - ~rl\_Ray, 50
  - attenuate, 50
  - derotate\_detranslate, 51
  - get\_PolCSPOD, 51
  - init\_ray, 51
  - intensity, 52
  - polarization, 52
  - print\_on, 52
  - reflect, 52
  - rl\_Ray, 50
  - rl\_Ray, 50
  - set\_polarization, 53
  - translate\_rotate, 53
- rl\_raylib/ Directory Reference, 3
- rl\_ReflectionCoefPOD, 54
  - cprint\_on, 54
  - init, 54, 55
  - para, 55
  - perp, 55, 56
  - print\_on, 56
  - reflectivity, 56
- rl\_Traits, 57
  - Bool, 58
  - complex, 58
  - EInterpMode, 58
  - ERoughType, 59
- rl\_Traits::rl\_DielectricPOD, 21
  - alpha\_, 22
  - energy\_, 22
  - gamma\_, 22
- rl\_TransmissionCoefPOD, 59
  - cprint\_on, 60
  - init, 60
  - para, 60, 61
  - perp, 61
  - print\_on, 61
  - transmission, 62
- rotate
  - rl\_Polarization, 44
- roughness
  - rl\_DielectricLayer, 20
- roughness\_type
  - rl\_DielectricLayer, 20

---

- s2\_
  - rl\_PolCSPOD, [48](#)
- S\_i
  - rl\_Polarization, [45](#)
- S\_r
  - rl\_Polarization, [45](#)
- set\_normal
  - rl\_MultilayerSurface, [39](#)
- set\_polarization
  - rl\_Ray, [53](#)
- setup\_for
  - rl\_DielectricLayer, [20](#)
- thickness
  - rl\_DielectricLayer, [21](#)
- translate\_rotate
  - rl\_Ray, [53](#)
- transmission
  - rl\_TransmissionCoefPOD, [62](#)
- vacuum\_
  - rl\_Multilayer, [35](#)
- zcoat
  - rl\_DielectricLayer, [21](#)