

rl\_raylib  
1.1.10

Generated by Doxygen 1.8.15



<b>1 rl_RayLib User's Guide</b>	<b>1</b>
1.1 Copyright and License	1
1.2 Purpose	1
1.2.1 Ray class	1
1.2.2 Reflectivity classes	2
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 rl_DielectricData Class Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 rl_DielectricData() [1/3]	6
3.1.2.2 rl_DielectricData() [2/3]	6
3.1.2.3 rl_DielectricData() [3/3]	6
3.1.3 Member Function Documentation	7
3.1.3.1 alpha_gamma()	7
3.1.3.2 bulk_density_factor()	7
3.1.3.3 init()	8
3.2 rl_DielectricData::rl_DielectricDataBinPOD Struct Reference	8
3.2.1 Detailed Description	9
3.3 rl_DielectricLayer Class Reference	9
3.3.1 Detailed Description	11
3.3.2 Constructor & Destructor Documentation	11
3.3.2.1 rl_DielectricLayer() [1/3]	11
3.3.2.2 rl_DielectricLayer() [2/3]	11
3.3.2.3 rl_DielectricLayer() [3/3]	12
3.3.3 Member Function Documentation	12
3.3.3.1 alpha()	12
3.3.3.2 bulk_density_factor()	13
3.3.3.3 cdump_on()	13
3.3.3.4 cprint_constraints_on()	13
3.3.3.5 dump_on()	14
3.3.3.6 energy_max()	14
3.3.3.7 energy_min()	14
3.3.3.8 gamma()	15
3.3.3.9 init()	15
3.3.3.10 is_substrate()	15
3.3.3.11 is_vacuum()	16

3.3.3.12 layer_name()	16
3.3.3.13 propagator()	16
3.3.3.14 reflect_amp()	16
3.3.3.15 reflect_nlayer()	17
3.3.3.16 reflection_coef()	17
3.3.3.17 reflectivity()	17
3.3.3.18 roughness()	18
3.3.3.19 roughness_type()	18
3.3.3.20 setup_for()	18
3.3.3.21 thickness()	19
3.3.3.22 zcoat()	19
3.4 rl_Traits::rl_DielectricPOD Struct Reference	19
3.4.1 Detailed Description	20
3.5 rl_DielectricPODArray Class Reference	20
3.5.1 Detailed Description	21
3.5.2 Constructor & Destructor Documentation	21
3.5.2.1 rl_DielectricPODArray() [1/3]	22
3.5.2.2 rl_DielectricPODArray() [2/3]	22
3.5.2.3 rl_DielectricPODArray() [3/3]	22
3.5.3 Member Function Documentation	23
3.5.3.1 const_data_ptr()	23
3.5.3.2 cprint_on()	23
3.5.3.3 init() [1/2]	24
3.5.3.4 init() [2/2]	24
3.5.3.5 num_elts()	24
3.6 rl_Exception Class Reference	25
3.6.1 Detailed Description	25
3.7 rl_Multilayer Class Reference	26
3.7.1 Detailed Description	27
3.7.2 Constructor & Destructor Documentation	27
3.7.2.1 ~rl_Multilayer()	27
3.7.2.2 rl_Multilayer()	27
3.7.3 Member Function Documentation	28
3.7.3.1 cdump_on()	28
3.7.3.2 dump_on()	28
3.7.3.3 init()	29
3.7.3.4 layer()	29
3.7.3.5 multilayer_reflect_coef()	30
3.7.3.6 multilayer_reflectivity()	30

3.7.3.7 num_layers()	31
3.8 rl_MultilayerSurface Class Reference	31
3.8.1 Detailed Description	32
3.8.2 Constructor & Destructor Documentation	32
3.8.2.1 rl_MultilayerSurface()	32
3.8.3 Member Function Documentation	33
3.8.3.1 dump_on()	33
3.8.3.2 layer()	33
3.8.3.3 normal_vector()	34
3.8.3.4 reflect()	34
3.8.3.5 reflection_coefs()	34
3.8.3.6 set_normal()	35
3.9 rl_Polarization Class Reference	35
3.9.1 Detailed Description	36
3.9.2 Constructor & Destructor Documentation	36
3.9.2.1 rl_Polarization() [1/2]	36
3.9.2.2 rl_Polarization() [2/2]	37
3.9.3 Member Function Documentation	37
3.9.3.1 attenuate()	37
3.9.3.2 cprint_on()	38
3.9.3.3 derotate()	38
3.9.3.4 get_PolCSPOD()	38
3.9.3.5 init()	39
3.9.3.6 intensity()	39
3.9.3.7 print_on()	39
3.9.3.8 reflect()	40
3.9.3.9 rotate()	40
3.10 rl_PolCSPOD Struct Reference	41
3.10.1 Detailed Description	41
3.10.2 Member Function Documentation	42
3.10.2.1 attenuate()	42
3.10.2.2 cprint_on()	42
3.10.2.3 init()	42
3.10.2.4 intensity()	43
3.10.2.5 print_on()	43
3.11 rl_Ray Class Reference	44
3.11.1 Detailed Description	44
3.11.2 Constructor & Destructor Documentation	45
3.11.2.1 rl_Ray() [1/3]	45

3.11.2.2 <code>rl_Ray()</code> [2/3]	45
3.11.2.3 <code>rl_Ray()</code> [3/3]	45
3.11.3 Member Function Documentation	46
3.11.3.1 <code>attenuate()</code>	46
3.11.3.2 <code>derotate_detranslate()</code>	46
3.11.3.3 <code>get_PolCSPOD()</code>	46
3.11.3.4 <code>init_ray()</code>	47
3.11.3.5 <code>intensity()</code>	47
3.11.3.6 <code>polarization()</code>	48
3.11.3.7 <code>print_on()</code>	48
3.11.3.8 <code>reflect()</code>	48
3.11.3.9 <code>set_polarization()</code>	49
3.11.3.10 <code>translate_rotate()</code>	49
3.12 <code>rl_ReflectionCoefPOD</code> Class Reference	49
3.12.1 Detailed Description	50
3.12.2 Member Function Documentation	50
3.12.2.1 <code>cprint_on()</code>	50
3.12.2.2 <code>init()</code>	51
3.12.2.3 <code>para()</code> [1/2]	51
3.12.2.4 <code>para()</code> [2/2]	51
3.12.2.5 <code>perp()</code> [1/2]	52
3.12.2.6 <code>perp()</code> [2/2]	52
3.12.2.7 <code>print_on()</code>	52
3.12.2.8 <code>reflectivity()</code>	53
3.13 <code>rl_Traits</code> Class Reference	53
3.13.1 Detailed Description	54
3.13.2 Member Enumeration Documentation	54
3.13.2.1 <code>EInterpMode</code>	54
3.13.2.2 <code>ERoughType</code>	55
3.14 <code>rl_TransmissionCoefPOD</code> Class Reference	55
3.14.1 Detailed Description	56
3.14.2 Member Function Documentation	56
3.14.2.1 <code>cprint_on()</code>	56
3.14.2.2 <code>init()</code>	56
3.14.2.3 <code>para()</code> [1/2]	57
3.14.2.4 <code>para()</code> [2/2]	57
3.14.2.5 <code>perp()</code> [1/2]	57
3.14.2.6 <code>perp()</code> [2/2]	57
3.14.2.7 <code>print_on()</code>	57

3.14.2.8 transmission()	58
-------------------------	----





# Chapter 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 `asubstrate`" (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.

## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">rl_DielectricData</a>	A class encapsulating the dielectric data (alpha, gamma) as a function of energy . . . . .	5
<a href="#">rl_DielectricData::rl_DielectricDataBinPOD</a>	A POD describing the lower and upper edge of an energy bin . . . . .	8
<a href="#">rl_DielectricLayer</a>	A class encapsulating the multilayer reflection of a ray . . . . .	9
<a href="#">rl_Traits::rl_DielectricPOD</a>	. . . . .	19
<a href="#">rl_DielectricPODArray</a>	A class encapsulating an array of rl_DielectricPODs storing the dielectric constants as a function of energy: . . . . .	20
<a href="#">rl_Exception</a>	The exception thrown by the rl_RayLib and rl_RaySupLib libraries . . . . .	25
<a href="#">rl_Multilayer</a>	A class encapsulating reflection of a ray from a multilayer surface . . . . .	26
<a href="#">rl_MultilayerSurface</a>	A class encapsulating a multilayer surface, including surface normal . . . . .	31
<a href="#">rl_Polarization</a>	Encapsulates the polarization state of a ray . . . . .	35
<a href="#">rl_PolCSPOD</a>	A Plain Old Data struct (POD) encapsulating the OSAC-style complex polarization amplitudes . . . . .	41
<a href="#">rl_Ray</a>	An rl_BasicRay with added polarization information . . . . .	44
<a href="#">rl_ReflectionCoeffPOD</a>	A Plain Old Data class representing complex reflection coefficients . . . . .	49
<a href="#">rl_Traits</a>	RL_Traits is a "traits" class for the rl_RayLib library . . . . .	53
<a href="#">rl_TransmissionCoeffPOD</a>	A Plain Old Data class representing complex reflection coefficients . . . . .	55



## Chapter 3

# Class Documentation

### 3.1 rl\_DielectricData Class Reference

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

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

#### Classes

- struct [rl\\_DielectricDataBinPOD](#)  
*A POD describing the lower and upper edge of an energy bin.*

#### Public Member Functions

- [~rl\\_DielectricData](#) ()  
*Non-virtual destructor.*
- [rl\\_DielectricData](#) ()  
*Default constructor.*
- [rl\\_DielectricData](#) ([rl\\_DielectricData](#) const &other)  
*Copy constructor.*
- [rl\\_DielectricData](#) ([rl\\_Traits::rl\\_DielectricPOD](#) const \*diel, size\_t num\_pts, [rl\\_Traits::EInterpMode](#) interp\_mode, double bulk\_density=1.0)  
*Constructor.*
- void [init](#) ([rl\\_Traits::rl\\_DielectricPOD](#) const \*diel, size\_t num\_pts, [rl\\_Traits::EInterpMode](#) interp\_mode, double bulk\_density=1.0)  
*Initialization function.*
- int [alpha\\_gamma](#) (double energy, double &alpha, double &gamma)  
*Evaluate the dielectric decrements, alpha and gamma, at the given energy.*
- double [energy\\_min](#) () const  
*Return the minimum energy covered by this dataset.*
- double [energy\\_max](#) () const  
*Return the maximum energy covered by this dataset.*
- double [bulk\\_density\\_factor](#) () const  
*Return the maximum energy covered by this dataset.*
- [rl\\_Traits::Bool](#) [is\\_vacuum](#) () const  
*Returns True if the object is a vacuum state; False, otherwise.*

### 3.1.1 Detailed Description

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

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

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 `rl_DielectricData()` [1/3]

```
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 212 of file `rl_DielectricData.h`.

#### 3.1.2.2 `rl_DielectricData()` [2/3]

```
rl_DielectricData::rl_DielectricData (
    rl\_DielectricData const & other )
```

Copy constructor.

##### Parameters

<i>other</i>	<a href="#">rl_DielectricData</a> to be copied.
--------------	---

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

#### 3.1.2.3 `rl_DielectricData()` [3/3]

```
rl_DielectricData::rl_DielectricData (
    rl\_Traits::rl\_DielectricPOD const * diel,
    size_t num_pts,
    rl\_Traits::EInterpMode interp_mode,
    double bulk_density = 1.0 )
```

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

<i>diel</i>	an array of <code>num_pts</code> POD's containing the complex dielectric decrements.
<i>num_pts</i>	number of elements in <i>diel</i> .
<i>interp_mode</i>	type of interpolation to be used in interpolating the dielectric constants.
<i>bulk_density</i>	relative bulk density factor to be used; 1.0 for full bulk density. The dielectric decrements will be scaled by <code>bulk_density</code> .

Definition at line 74 of file `rl_DielectricData.cc`.

### 3.1.3 Member Function Documentation

#### 3.1.3.1 alpha\_gamma()

```
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

<i>energy</i>	photon energy (in keV).
<i>alpha</i>	real part of the complex dielectric decrement.
<i>gamma</i>	imaginary part of the complex dielectric decrement.

Definition at line 173 of file `rl_DielectricData.cc`.

#### 3.1.3.2 bulk\_density\_factor()

```
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 236 of file rl\_DielectricData.h.

### 3.1.3.3 init()

```
void rl_DielectricData::init (
    rl_Traits::rl_DielectricPOD const * diel,
    size_t num_pts,
    rl_Traits::EInterpMode interp_mode,
    double bulk_density = 1.0 )
```

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

<i>diel</i>	an array of num_pts POD's containing the complex dielectric decrements.
<i>num_pts</i>	number of elements in diel.
<i>interp_mode</i>	type of interpolation to be used in interpolating the dielectric constants.
<i>bulk_density</i>	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 84 of file rl\_DielectricData.cc.

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

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

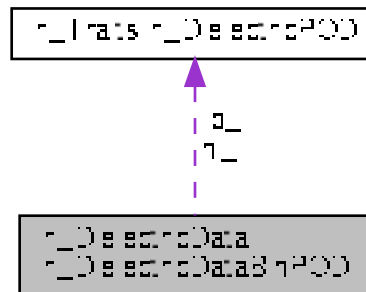
## 3.2 rl\_DielectricData::rl\_DielectricDataBinPOD Struct Reference

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

```
#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*

### 3.2.1 Detailed Description

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

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

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

- rl\_DielectricData.h

## 3.3 rl\_DielectricLayer Class Reference

A class encapsulating the multilayer reflection of a ray.

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

### Public Types

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

## Public Member Functions

- [~rl\\_DielectricLayer](#) ()  
*Destructor.*
- [rl\\_DielectricLayer](#) (char const [layer\\_name](#)[]=0)  
*Constructor.*
- [rl\\_DielectricLayer](#) ([rl\\_DielectricLayer](#) const &other)  
*DEEP Copy constructor.*
- [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](#))  
*Constructor.*
- 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](#))  
*Initializer.*
- int [setup\\_for](#) (double energy, double sinphi)  
*Set up layer state for given energy and graze angle.*
- void [reflect\\_nlayer](#) ([rl\\_DielectricLayer](#) layer[], int num)  
*Compute reflectivity for a stack of num layers.*
- void [reflect\\_amp](#) ([rl\\_DielectricLayer](#) const &layer, double sinphi)  
*Compute reflection amplitude for the interface between this layer and the layer immediately above it.*
- [complex](#) const & [propagator](#) () const  
*The propagator for this layer.*
- double [alpha](#) () const  
*Returns this layer's dielectric decrement (real part).*
- double [gamma](#) () const  
*Returns this layer's dielectric decrement (imag part).*
- double [roughness](#) () const  
*Returns the roughness parameter of the upper surface of this layer.*
- [ERoughType](#) [roughness\\_type](#) () const  
*Returns the roughness type of the upper surface of this layer.*
- [rl\\_ReflectionCoefPOD](#) const & [reflection\\_coef](#) () const  
*Returns this layer's complex reflection coefficient.*
- double [reflectivity](#) (double polarization\_factor=0.0) const  
*Returns: this layer's reflectivity.*
- [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  
*Dumps layer information to a stream.*
- void [cdump\\_on](#) (std::FILE \*of, char const pre[]="", char const pst[]="") const  
*Dumps layer information to a C-style FILE\* stream.*
- void [cprint\\_constraints\\_on](#) (std::FILE \*of, char const pre[]="", char const pst[]="") const  
*Dumps layer information and constraints to a C-style FILE\* stream.*

### 3.3.1 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.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 rl\_DielectricLayer() [1/3]

```
rl_DielectricLayer::rl_DielectricLayer (
    char const layer_name[ ] = 0 )
```

Constructor.

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

Parameters

<i>layer_name</i>	optional string naming the layer.
-------------------	-----------------------------------

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

#### 3.3.2.2 rl\_DielectricLayer() [2/3]

```
rl_DielectricLayer::rl_DielectricLayer (
    rl_DielectricLayer const & other )
```

DEEP Copy constructor.

Parameters

<i>other</i>	rl_DielectricLayer to be copied.
--------------	----------------------------------

Definition at line 98 of file rl\_DielectricLayer.cc.

### 3.3.2.3 rl\_DielectricLayer() [3/3]

```
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 )
```

Constructor.

#### Parameters

<i>diel</i>	array of dielectric decrement information.
<i>ndielpts</i>	number of points in diel array.
<i>layer_thickness</i>	layer thickness (Angstroms).
<i>roughness</i>	layer roughness ' ' (Angstroms). @param roughness_type interlayer gradingroughness" type.
<i>interp_mode</i>	type of optical constant interpolation to be done.
<i>bulkdensity</i>	relative bulk density factor; 1 for full bulk density.
<i>layer_name</i>	string describing the layer composition.
<i>is_substrate</i>	boolean: True if this is the substrate layer.

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

## 3.3.3 Member Function Documentation

### 3.3.3.1 alpha()

```
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 524 of file rl\_DielectricLayer.h.

### 3.3.3.2 bulk\_density\_factor()

```
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 576 of file rl\_DielectricLayer.h.

### 3.3.3.3 cdump\_on()

```
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

<i>of</i>	output file.
<i>pre</i>	optional prefix string.
<i>pst</i>	optional postfix string.

Definition at line 832 of file rl\_DielectricLayer.cc.

### 3.3.3.4 cprint\_constraints\_on()

```
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

<i>of</i>	output file.
<i>pre</i>	optional prefix string.
<i>pst</i>	optional postfix string.

Definition at line 876 of file rl\_DielectricLayer.cc.

#### 3.3.3.5 dump\_on()

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

Dumps layer information to a stream.

##### Parameters

<i>os</i>	stream.
<i>pre</i>	optional prefix string.
<i>pst</i>	optional postfix string.

Definition at line 787 of file rl\_DielectricLayer.cc.

#### 3.3.3.6 energy\_max()

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

##### Returns

the maximum energy allowed for this layer.

Definition at line 572 of file rl\_DielectricLayer.h.

#### 3.3.3.7 energy\_min()

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

##### Returns

the minimum energy allowed for this layer.

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

## 3.3.3.8 gamma()

```
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 528 of file rl\_DielectricLayer.h.

## 3.3.3.9 init()

```
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 )
```

Initializer.

## Parameters

<i>diel</i>	array of dielectric decrement information.
<i>ndielpts</i>	number of points in diel array.
<i>layer_thickness</i>	layer thickness (Angstroms).
<i>roughness</i>	interlayer grading "roughness" (Angstrom).
<i>roughness_type</i>	interlayer grading "roughness" type.
<i>interp_mode</i>	type of optical constant interpolation to be done.
<i>bulkdensity</i>	relative bulk density factor; 1 for full bulk density.
<i>layer_name</i>	string describing the layer composition.
<i>is_substrate</i>	boolean: True if this is the substrate layer.

Definition at line 157 of file rl\_DielectricLayer.cc.

## 3.3.3.10 is\_substrate()

```
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 556 of file `rl_DielectricLayer.h`.

**3.3.3.11 is\_vacuum()**

```
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 560 of file `rl_DielectricLayer.h`.

**3.3.3.12 layer\_name()**

```
char const * rl_DielectricLayer::layer_name ( ) const [inline]
```

**Returns**

the name of this layer as a character string.

Definition at line 564 of file `rl_DielectricLayer.h`.

**3.3.3.13 propagator()**

```
rl_Traits::complex const & rl_DielectricLayer::propagator ( ) const [inline]
```

The propagator for this layer.

**Returns**

propagator for this layer.

Definition at line 520 of file `rl_DielectricLayer.h`.

**3.3.3.14 reflect\_amp()**

```
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

<i>layer</i>	adjacent layer above this layer, where above means closer to the vacuum.
<i>sinphi</i>	sine of the graze angle between the ray (in vacuum) and the top surface of the multilayer.

Definition at line 320 of file rl\_DielectricLayer.cc.

## 3.3.3.15 reflect\_nlayer()

```
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

<i>layer</i>	array of layers.
<i>num</i>	number of layers in the array.

Definition at line 516 of file rl\_DielectricLayer.cc.

## 3.3.3.16 reflection\_coef()

```
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 548 of file rl\_DielectricLayer.h.

## 3.3.3.17 reflectivity()

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

Returns: this layer's reflectivity.

**Parameters**

<i>polarization_factor</i>	- 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 552 of file rl\_DielectricLayer.h.

**3.3.3.18 roughness()**

```
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 544 of file rl\_DielectricLayer.h.

**3.3.3.19 roughness\_type()**

```
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 540 of file rl\_DielectricLayer.h.

**3.3.3.20 setup\_for()**

```
int rl_DielectricLayer::setup_for (
    double energy,
    double sinphi )
```

Set up layer state for given energy and graze angle.

**Parameters**

<i>energy</i>	energy (keV).
<i>sinphi</i>	sine of the graze angle between the ray (in vacuum) and the top surface of the multilayer.

Definition at line 220 of file rl\_DielectricLayer.cc.

**3.3.3.21 thickness()**

```
double rl_DielectricLayer::thickness ( ) const [inline]
```

**Returns**

the layer thickness.

Definition at line 532 of file rl\_DielectricLayer.h.

**3.3.3.22 zcoat()**

```
double rl_DielectricLayer::zcoat ( ) const [inline]
```

**Returns**

the layer dimensionless thickness.

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

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

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

**3.4 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*

### 3.4.1 Detailed Description

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

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

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

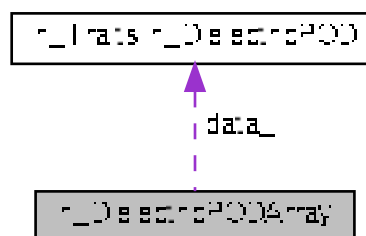
- rl\_Traits.h

## 3.5 rl\_DielectricPODArray Class Reference

A class encapsulating an array of rl\_DielectricPODs storing the dielectric constants as a function of energy:

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

Collaboration diagram for rl\_DielectricPODArray:



## Public Member Functions

- [~rl\\_DielectricPODArray](#) ()  
*Destructor.*
- [rl\\_DielectricPODArray](#) ()  
*Default constructor.*
- [rl\\_DielectricPODArray](#) (size\_t nelts, double const \*energy, double const \*alpha, double const \*gamma)  
*Constructor.*
- [rl\\_DielectricPODArray](#) (size\_t nelts, [rl\\_Traits::rl\\_DielectricPOD](#) \*diel)  
*Constructor.*
- void [init](#) (size\_t nelts, double const \*energy, double const \*alpha, double const \*gamma)  
*Initializer.*
- void [init](#) (size\_t nelts, [rl\\_Traits::rl\\_DielectricPOD](#) \*diel)  
*Initializer.*
- size\_t [num\\_elts](#) () const  
*Accessor.*
- [rl\\_Traits::rl\\_DielectricPOD](#) const \* [const\\_data\\_ptr](#) () const  
*Accessor.*
- void [cprint\\_on](#) (std::FILE \*of, char const pre[]="", char const pst[]="") const  
*Accessor.*

## Protected Attributes

- size\_t [nelts\\_](#)  
*number of dielectric decrements read in*
- [rl\\_Traits::rl\\_DielectricPOD](#) \* [data\\_](#)  
*pointer to the data*

### 3.5.1 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](#).

### 3.5.2 Constructor & Destructor Documentation

### 3.5.2.1 `rl_DielectricPODArray()` [1/3]

```
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`.

### 3.5.2.2 `rl_DielectricPODArray()` [2/3]

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

Constructor.

#### Parameters

<i>nelts</i>	number of elements in the array
<i>energy</i>	array of energies
<i>alpha</i>	array of dielectric decrement real part (alpha)
<i>gamma</i>	array of dielectric decrement imag part (gamma)

Definition at line 74 of file `rl_DielectricPODArray.cc`.

### 3.5.2.3 `rl_DielectricPODArray()` [3/3]

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

Constructor.

#### Parameters

<i>nelts</i>	number of elements in the array
<i>diel</i>	array of dielectric decrement PODs

Definition at line 100 of file rl\_DielectricPODArray.cc.

### 3.5.3 Member Function Documentation

#### 3.5.3.1 const\_data\_ptr()

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

Accessor.

##### Returns

pointer-to-const to rl\_DielectricPOD array.

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

#### 3.5.3.2 cprint\_on()

```
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

<i>of</i>	output FILE* stream.
<i>pre</i>	optional prefix (char*) string.
<i>pst</i>	optional postfix (char*) string.

Definition at line 154 of file rl\_DielectricPODArray.cc.

**3.5.3.3** `init()` [1/2]

```
void rl_DielectricPODArray::init (
    size_t nelts,
    double const * energy,
    double const * alpha,
    double const * gamma )
```

Initializer.

**Parameters**

<i>nelts</i>	number of elements in the array
<i>energy</i>	array of energies
<i>alpha</i>	array of dielectric decrement real part (alpha)
<i>gamma</i>	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 120 of file `rl_DielectricPODArray.cc`.

**3.5.3.4** `init()` [2/2]

```
void rl_DielectricPODArray::init (
    size_t nelts,
    rl_Traits::rl_DielectricPOD * diel )
```

Initializer.

**Parameters**

<i>nelts</i>	number of elements in the array
<i>diel</i>	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 138 of file `rl_DielectricPODArray.cc`.

**3.5.3.5** `num_elts()`

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

Accessor.



#### Returns

number of elements in the rl\_DielectricPOD array.

Definition at line 168 of file rl\_DielectricPODArray.h.

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

- rl\_DielectricPODArray.h
- rl\_DielectricPODArray.cc

## 3.6 rl\_Exception Class Reference

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

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

Inherits Exception.

### Public Member Functions

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

### 3.6.1 Detailed Description

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

Definition at line 36 of file rl\_Exception.h.

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

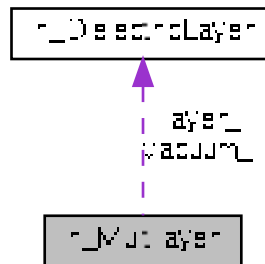
- rl\_Exception.h
- rl\_Exception.cc

### 3.7 rl\_Multilayer Class Reference

A class encapsulating reflection of a ray from a multilayer surface.

```
#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](#) ()  
*Destructor.*
- [rl\\_Multilayer](#) (int [num\\_layers](#), [rl\\_DielectricLayer](#) \*layers, [rl\\_Traits::Bool](#) adopt\_data=[rl\\_Traits::True](#))  
*Construct multilayer from num\_layers individual layers.*
- void [init](#) (int [num\\_layers](#), [rl\\_DielectricLayer](#) \*layers, [rl\\_Traits::Bool](#) adopt\_data=[rl\\_Traits::True](#))  
*Initialize multilayer from num\_layers individual layers.*
- int [multilayer\\_reflect\\_coef](#) ([rl\\_ReflectionCoefPOD](#) &rfl, double energy, double sg)  
*Evaluate the multilayer reflection coefficients.*
- int [multilayer\\_reflectivity](#) (double &rfl, double energy, double sg, double polarization\_factor=0.0)  
*Evaluate the multilayer reflectivity (assuming unpolarized rays)*
- [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  
*Dump information about layer layer\_no to stream os.*
- void [cdump\\_on](#) (std::FILE \*of, int layer\_no, char const pre[ ]="", char const pst[ ]="") const  
*Dump information about layer layer\_no to output FILE\*.*

## 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?*

### 3.7.1 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`.

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 ~rl\_Multilayer()

```
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`.

#### 3.7.2.2 rl\_Multilayer()

```
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

<i>num_layers</i>	number of layers to construct.
<i>layers</i>	pointer to array of layer data
<i>adopt_data</i>	boolean flag <ul style="list-style-type: none"> <li>• if <code>rl_Traits::True</code>, <code>rl_Multilayer</code> assumes responsibility for deleting the <code>rl_DielectricLayer</code> array;</li> <li>• if <code>rl_Traits::False</code>, it is the caller's responsibility.</li> </ul>

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

### 3.7.3 Member Function Documentation

#### 3.7.3.1 `cdump_on()`

```
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

<i>of</i>	output <code>FILE*</code>
<i>layer_no</i>	layer number to be dumped
<i>pre</i>	optional prefix string
<i>pst</i>	optional postfix string

Definition at line 181 of file `rl_Multilayer.cc`.

#### 3.7.3.2 `dump_on()`

```
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**

<i>os</i>	output stream
<i>layer_no</i>	layer number to be dumped
<i>pre</i>	optional prefix string
<i>pst</i>	optional postfix string

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

**3.7.3.3 init()**

```
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**

<i>num_layers</i>	number of layers to construct.
<i>layers</i>	pointer to array of layer data
<i>adopt_data</i>	boolean flag <ul style="list-style-type: none"> <li>if rl_Traits::True, rl_Multilayer assumes responsibility for deleting the rl_DielectricLayer array;</li> <li>if rl_Traits::False, it is the caller's responsibility.</li> </ul>

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

**3.7.3.4 layer()**

```
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.

### 3.7.3.5 multilayer\_reflect\_coef()

```
int rl_Multilayer::multilayer_reflect_coef (
    rl_ReflectionCoefPOD & rfl,
    double energy,
    double sg )
```

Evaluate the multilayer reflection coefficients.

#### Returns

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

#### Parameters

<i>rfl</i>	the computed reflection coefficient.
<i>energy</i>	ray energy.
<i>sg</i>	sine of the graze angle between the ray and the surface.

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

### 3.7.3.6 multilayer\_reflectivity()

```
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

<i>rfl</i>	multilayer reflectivity.
<i>energy</i>	ray energy.
<i>sg</i>	sine of the graze angle between the ray and the surface.

## Parameters

<i>polarization_factor</i>	<p>polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:</p> $\text{polarization\_factor} = \frac{(I_{\perp} - I_{\parallel})}{(I_{\perp} + I_{\parallel})}$ <p>or</p> $\text{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$ <p>where <math>I_{\perp}</math> and <math>I_{\parallel}</math> are the perpendicular and parallel E-field <i>intensities</i>, respectively. Thus,</p> <ul style="list-style-type: none"> <li>• -1: pure parallel (p) polarization.</li> <li>• 0: completely unpolarized.</li> <li>• +1: pure perpendicular (s) polarization.</li> </ul>
----------------------------	--

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

## 3.7.3.7 num\_layers()

```
int rl_Multilayer::num_layers ( ) const [inline]
```

## Returns

number of layers

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

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

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

## 3.8 rl\_MultilayerSurface Class Reference

A class encapsulating a multilayer surface, including surface normal.

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

## Public Member Functions

- virtual [~rl\\_MultilayerSurface](#) ()  
*Destructor.*
- [rl\\_MultilayerSurface](#) ([rl\\_Multilayer](#) &ml, [dvm3\\_Vector](#) &norm)  
*Constructor.*
- [dvm3\\_Vector](#) const & [normal\\_vector](#) () const  
*Accessor.*
- void [set\\_normal](#) ([dvm3\\_Vector](#) const &norm)  
*Mutator.*
- int [reflect](#) ([rl\\_Ray](#) &ray)  
*Mutator.*
- [rl\\_ReflectionCoefPOD](#) const & [reflection\\_coefs](#) () const  
*Accessor.*
- [rl\\_DielectricLayer](#) const & [layer](#) (int layer\_no) const  
*Accessor.*
- std::ostream & [dump\\_on](#) (std::ostream &os, int layer\_no, char const pre[ ]="", char const pst[ ]="") const  
*Dump information about a layer.*

### 3.8.1 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](#).

### 3.8.2 Constructor & Destructor Documentation

#### 3.8.2.1 [rl\\_MultilayerSurface](#)()

```
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

<i>ml</i>	the <a href="#">rl_Multilayer</a> object to be used (must already exist)
<i>norm</i>	the surface normal <a href="#">dvm3_Vector</a> to be used (must already exist)



Definition at line 181 of file rl\_MultilayerSurface.h.

### 3.8.3 Member Function Documentation

#### 3.8.3.1 dump\_on()

```
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

<i>os</i>	output stream
<i>layer_no</i>	layer number for which the information is requested.
<i>pre</i>	optional char* prefix string.
<i>pst</i>	optional char* postfix string.

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

#### 3.8.3.2 layer()

```
rl_DielectricLayer const & rl_MultilayerSurface::layer (
    int layer_no ) const [inline]
```

Accessor.

##### Parameters

<i>layer_no</i>	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.

### 3.8.3.3 normal\_vector()

```
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.

### 3.8.3.4 reflect()

```
int rl_MultilayerSurface::reflect (
    rl_Ray & ray ) [inline]
```

Mutator.

Reflect ray direction vector at this surface

#### Parameters

<i>ray</i>	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.

### 3.8.3.5 reflection\_coefs()

```
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.

## 3.8.3.6 set\_normal()

```
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

<i>norm</i>	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

## 3.9 rl\_Polarization Class Reference

Encapsulates the polarization state of a ray.

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

## Public Types

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

## Public Member Functions

- [~rl\\_Polarization](#) ()  
*Destructor (NON-VIRTUAL)*
- [rl\\_Polarization](#) ()  
*Default constructor; constructs [rl\\_Polarization](#) with INVALID fields.*
- [rl\\_Polarization](#) ([rl\\_Polarization](#) const &rhs)  
*Copy constructor; constructs [rl\\_Polarization](#) with INVALID fields.*
- [rl\\_Polarization](#) ([rl\\_PolCSPD](#) const &cs, [dvm3\\_Vector](#) const &dir)  
*Conversion constructor.*
- void [init](#) ([rl\\_PolCSPD](#) const &cs, [dvm3\\_Vector](#) const &dir)

- Initialize the polarization state.*
- void `get_PolCSPOD` (`rl_PolCSPOD` &cs, `dvm3_Vector` const &dir) const  
*Evaluate `rl_PolCSPOD` corresponding to this polarization state.*
- double `intensity` () const  
*Evaluate intensity.*
- void `rotate` (`rl_Polarization` &rotated, `dvm3_RotMat` const &rot\_mtx)  
*Rotate polarization state to frame described by `rot_mtx`.*
- void `derotate` (`rl_Polarization` &derotated, `dvm3_RotMat` const &rot\_mtx)  
*Rotate polarization state back from frame described by `rot_mtx`.*
- void `attenuate` (double factor)  
*Attenuate reflectivity by a factor.*
- void `reflect` (`dvm3_Vector` const &normal, `dvm3_Vector` const &dir\_in, `dvm3_Vector` const &dir\_out, `rl_ReflectionCoefPOD` const &rflcoef)  
*Evaluate the reflected polarization vectors.*
- `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  
*Write the polarization vectors to stream os.*
- void `cprint_on` (`std::FILE` \*of) const  
*Write the polarization amplitude to FILE\* of.*

### 3.9.1 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`.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 `rl_Polarization()` [1/2]

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

Copy constructor; constructs `rl_Polarization` with INVALID fields.

## Parameters

<i>rhs</i>	- <a href="#">rl_Polarization</a> object to be copied.
------------	--

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

3.9.2.2 `rl_Polarization()` [2/2]

```
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

<i>cs</i>	OSAC-style polarization amplitude
<i>dir</i>	ray direction vector

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

## 3.9.3 Member Function Documentation

3.9.3.1 `attenuate()`

```
void rl_Polarization::attenuate (
    double factor )
```

Attenuate reflectivity by a factor.

## Parameters

<i>factor</i>	attenuation factor. (NOTE: this multiplies the polarization components by <code>sqrt(factor)</code> .)
---------------	--

Definition at line 243 of file `rl_Polarization.cc`.

### 3.9.3.2 cprint\_on()

```
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

<i>of</i>	output FILE*.
-----------	---------------

Definition at line 350 of file rl\_Polarization.cc.

### 3.9.3.3 derotate()

```
void rl_Polarization::derotate (
    rl_Polarization & derotated,
    dvm3_RotMat const & rot_mtx ) [inline]
```

Rotate polarization state back from frame described by rot\_mtx.

#### Parameters

<i>derotated</i>	output derotated polarization state. The state is now back in the original frame.
<i>rot_mtx</i>	rotation matrix.

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

### 3.9.3.4 get\_PolCSPOD()

```
void rl_Polarization::get_PolCSPOD (
    rl_PolCSPOD & cs,
    dvm3_Vector const & dir ) const
```

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

#### Parameters

<i>cs</i>	OSAC-style polarization amplitude
<i>dir</i>	ray direction vector

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

#### 3.9.3.5 init()

```
void rl_Polarization::init (
    rl_PolCSPOD const & cs,
    dvm3_Vector const & dir )
```

Initialize the polarization state.

##### Parameters

<i>cs</i>	OSAC-style polarization amplitude
<i>dir</i>	ray direction vector

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

#### 3.9.3.6 intensity()

```
double rl_Polarization::intensity ( ) const
```

Evaluate intensity.

##### Returns

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

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

#### 3.9.3.7 print\_on()

```
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**

<i>os</i>	output stream.
-----------	----------------

Definition at line 339 of file `rl_Polarization.cc`.

**3.9.3.8 reflect()**

```
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**

<i>normal</i>	surface normal vector.
<i>dir_in</i>	direction vector for incoming ray.
<i>dir_out</i>	direction vector for reflected ray.
<i>rflcoef</i>	complex reflection coefficients for surface.

Definition at line 262 of file `rl_Polarization.cc`.

**3.9.3.9 rotate()**

```
void rl_Polarization::rotate (
    rl_Polarization & rotated,
    dvm3_RotMat const & rot_mtx ) [inline]
```

Rotate polarization state to frame described by `rot_mtx`.

**Parameters**

<i>rotated</i>	output rotated polarization state. The state is now in the frame obtained by applying <code>rot_mtx</code> .
<i>rot_mtx</i>	rotation matrix.

Definition at line 368 of file `rl_Polarization.h`.



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

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

## 3.10 rl\_PolCSPD Struct Reference

A Plain Ol' Data struct (POD) encapsulating the OSAC-style complex polarization amplitudes.

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

### Public Member Functions

- void **init** (double b\_over\_a=1.0, double psi=0.0)  
*Initialization method.*
- void **attenuate** (double factor)  
*Attenuate intensity by a factor.*
- double **intensity** () const  
*Evaluate the intensity.*
- std::ostream & **print\_on** (std::ostream &os) const  
*Write the polarization amplitude to stream os.*
- void **cprint\_on** (FILE \*of) const  
*Write the polarization amplitude to FILE\* of.*

### Public Attributes

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

#### 3.10.1 Detailed Description

A Plain Ol' 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:

$$c2\_q1 \iff c2comp(1), \quad c2\_q2 \iff c2comp(2) \quad s2\_q1 \iff s2comp(1), \quad s2\_q2 \iff s2comp(2)$$

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:

$$wt = c2\_q1.r^2 + c2\_q2.r^2 + c2\_q1.i^2 + c2\_q2.i^2 + s2\_q1.r^2 + s2\_q2.r^2 + s2\_q1.i^2 + s2\_q2.i^2 + 2(c2\_q1.is2\_q1.r + c2\_q2.ic2\_q2.r - c2\_q1.is2\_q2.r - c2\_q2.ic2\_q1.r)$$

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.

### 3.10.2 Member Function Documentation

#### 3.10.2.1 `attenuate()`

```
void rl_PolCSPD::attenuate (
    double factor )
```

Attenuate intensity by a factor.

##### Parameters

<i>factor</i>	the attenuation factor; the ray intensity is multiplied by factor.
---------------	--

Definition at line 121 of file `rl_Polarization.cc`.

#### 3.10.2.2 `cprint_on()`

```
void rl_PolCSPD::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

<i>of</i>	output <code>FILE*</code> .
-----------	-----------------------------

Definition at line 149 of file `rl_Polarization.cc`.

#### 3.10.2.3 `init()`

```
void rl_PolCSPD::init (
    double b_over_a = 1.0,
    double psi = 0.0 )
```

Initialization method.

## Parameters

<i>b_over_a</i>	ratio of semiminor axis to semi-major axis.
<i>psi</i>	angle of the major axis from the +X axis, with psi increasing towards +Y

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

## 3.10.2.4 intensity()

```
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.

## 3.10.2.5 print\_on()

```
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

<i>os</i>	output stream.
-----------	----------------

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

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

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

### 3.11 rl\_Ray Class Reference

An rl\_BasicRay with added polarization information.

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

Inherits rl\_BasicRay.

#### Public Member Functions

- virtual [~rl\\_Ray](#) ()  
*Destructor.*
- [rl\\_Ray](#) ()  
*Default constructor.*
- [rl\\_Ray](#) (rl\_BasicRay const &ray, [rl\\_PolCSPOD](#) const &cspol)  
*Constructor.*
- [rl\\_Ray](#) (dvm3\_Vector const &pos, dvm3\_Vector const &dir, double energy, long int id, [rl\\_PolCSPOD](#) const &cspol)  
*Constructor.*
- void [init\\_ray](#) (dvm3\_Vector const &pos, dvm3\_Vector const &dir, double energy, long int id, [rl\\_PolCSPOD](#) const &cspol)  
*Initialize a ray.*
- [rl\\_Polarization](#) const & [polarization](#) () const  
*Return the polarization state.*
- void [set\\_polarization](#) ([rl\\_PolCSPOD](#) const &cspol)  
*Set the polarization state.*
- void [get\\_PolCSPOD](#) ([rl\\_PolCSPOD](#) &cs) const  
*Evaluate [rl\\_PolCSPOD](#) corresponding to this ray's polarization state.*
- double [intensity](#) () const  
*Returns this ray's normalized intensity (i.e., "weight")*
- void [attenuate](#) (double by\_how\_much)  
*Attenuate this ray by by\_how\_much.*
- void [reflect](#) (dvm3\_Vector const &normal, [rl\\_ReflectionCoefPOD](#) const &rflcoef)  
*Reflect this ray's direction vector and polarization state.*
- void [translate\\_rotate](#) (dvm3\_Vector const &trans, dvm3\_RotMat const &rotmat)  
*Translate to BCS origin and rotate from STD to BCS coordinates.*
- void [derotate\\_detranslate](#) (dvm3\_Vector const &trans, dvm3\_RotMat const &rotmat)  
*Derotate back to std coordinates; detranslate back to std origin.*
- std::ostream & [print\\_on](#) (std::ostream &os, char const pre[]="", char const pst[]="") const  
*Write the ray contents with optional pre and post comment strings.*

#### 3.11.1 Detailed Description

An rl\_BasicRay with added polarization information.

Definition at line 45 of file rl\_Ray.h.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 rl\_Ray() [1/3]

```
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.

3.11.2.2 rl\_Ray() [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.

3.11.2.3 rl\_Ray() [3/3]

```
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

<i>pos</i>	ray position vector.
<i>dir</i>	ray direction unit vector.
<i>energy</i>	ray energy (keV).
<i>id</i>	ray numeric ray identifier.
<i>cspol</i>	an OSAC-style complex polarization amplitude vector.

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

### 3.11.3 Member Function Documentation

#### 3.11.3.1 attenuate()

```
void rl_Ray::attenuate (
    double by_how_much ) [inline]
```

Attenuate this ray by *by\_how\_much*.

##### Parameters

<i>by_how_much</i>	how much to attenuate this ray.
--------------------	---------------------------------

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

#### 3.11.3.2 derotate\_detranslate()

```
void rl_Ray::derotate_detranslate (
    dvm3_Vector const & trans,
    dvm3_RotMat const & rotmat )
```

Derotate back to std coordinates; detranslate back to std origin.

##### Parameters

<i>trans</i>	translation vector.
<i>rotmat</i>	rotation matrix to be applied. <i>rotmat</i> specifies a rotation from std to bcs; the inverse of <i>rotmat</i> is applied to the ray.

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

#### 3.11.3.3 get\_PolCSPD()

```
void rl_Ray::get_PolCSPD (
    rl_PolCSPD & cs ) const [inline]
```

Evaluate *rl\_PolCSPD* corresponding to this ray's polarization state.

## Parameters

<i>cs</i>	return an OSAC-style complex polarization amplitude vector.
-----------	---

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

## 3.11.3.4 init\_ray()

```
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

<i>pos</i>	ray position vector.
<i>dir</i>	ray direction unit vector.
<i>energy</i>	ray energy (keV).
<i>id</i>	a numeric ray identifier.
<i>cspol</i>	an OSAC-style complex polarization amplitude vector.

Definition at line 211 of file rl\_Ray.h.

## 3.11.3.5 intensity()

```
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.

### 3.11.3.6 polarization()

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

Return the polarization state.

#### Returns

polarization state

Definition at line 224 of file rl\_Ray.h.

### 3.11.3.7 print\_on()

```
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

<i>os</i>	output stream.
<i>pre</i>	optional prefix c-string.
<i>pst</i>	optional postfix c-string.

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

### 3.11.3.8 reflect()

```
void rl_Ray::reflect (
    dvm3_Vector const & normal,
    rl_ReflectionCoefPOD const & rflcoef )
```

Reflect this ray's direction vector and polarization state.

#### Parameters

<i>normal</i>	surface normal unit vector.
<i>rflcoef</i>	complex reflectances for the surface.



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

### 3.11.3.9 set\_polarization()

```
void rl_Ray::set_polarization (
    rl_PolCSPOD const & cspol ) [inline]
```

Set the polarization state.

#### Parameters

<i>cspol</i>	an OSAC-style complex polarization amplitude vector.
--------------	--

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

### 3.11.3.10 translate\_rotate()

```
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

<i>trans</i>	translation vector
<i>rotmat</i>	rotation matrix to be applied. rotmat specifies a rotation from std to bcs.

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

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

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

## 3.12 rl\_ReflectionCoefPOD Class Reference

A Plain Ol' Data class representing complex reflection coefficients.

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

## Public Member Functions

- double [reflectivity](#) (double polarization\_factor=0.0) const  
*evaluate the reflectivity.*
- void [init](#) ()  
*initialize perpendicular (s) and parallel (p) reflection coefficients to zero.*
- void [init](#) (rl\_Traits::complex &para, rl\_Traits::complex &perp)  
*initialize perpendicular (s) and parallel (p) reflection coefficients.*
- 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  
*Print reflectivity information to output stream.*
- void [cprint\\_on](#) (std::FILE \*of, char const pre[]="", char const pst[]="") const  
*Print reflectivity information to output FILE\* stream.*

### 3.12.1 Detailed Description

A Plain Ol' Data class representing complex reflection coefficients.

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

### 3.12.2 Member Function Documentation

#### 3.12.2.1 cprint\_on()

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

Print reflectivity information to output FILE\* stream.

#### Parameters

<i>of</i>	output FILE* stream.
<i>pre</i>	optional prefix (char*) string.
<i>pst</i>	optional postfix (char*) string.

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

### 3.12.2.2 init()

```
void rl_ReflectionCoefPOD::init (
    rl_Traits::complex & para,
    rl_Traits::complex & perp ) [inline]
```

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

to zero.

#### Parameters

<i>para</i>	parallel reflection coefficient.
<i>perp</i>	perpendicular reflection coefficient.

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

### 3.12.2.3 para() [1/2]

```
rl_Traits::complex rl_ReflectionCoefPOD::para ( ) const [inline]
```

#### Returns

parallel (p) reflection coefficient.

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

### 3.12.2.4 para() [2/2]

```
rl_Traits::complex & rl_ReflectionCoefPOD::para ( ) [inline]
```

#### Returns

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

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

### 3.12.2.5 `perp()` [1/2]

```
rl_Traits::complex rl_ReflectionCoefPOD::perp ( ) const [inline]
```

#### Returns

perpendicular (p) reflection coefficient.

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

### 3.12.2.6 `perp()` [2/2]

```
rl_Traits::complex & rl_ReflectionCoefPOD::perp ( ) [inline]
```

#### Returns

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

Definition at line 199 of file `rl_ReflectionCoefPOD.h`.

### 3.12.2.7 `print_on()`

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

Print reflectivity information to output stream.

#### Parameters

<i>os</i>	output stream.
<i>pre</i>	optional prefix (char*) string.
<i>pst</i>	optional postfix (char*) string.

Definition at line 210 of file `rl_ReflectionCoefPOD.h`.

## 3.12.2.8 reflectivity()

```
double rl_ReflectionCoefPOD::reflectivity (
    double polarization_factor = 0.0 ) const [inline]
```

evaluate the reflectivity.

## Parameters

<i>polarization_factor</i>	<p>polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:</p> $\text{polarization\_factor} = \frac{(I_{\perp} - I_{\parallel})}{(I_{\perp} + I_{\parallel})}$ <p>or</p> $\text{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$ <p>where <math>I_{\perp}</math> and <math>I_{\parallel}</math> are the perpendicular and parallel E-field <i>intensities</i>, respectively. Thus,</p> <ul style="list-style-type: none"> <li>• -1: pure parallel (p) polarization.</li> <li>• 0: completely unpolarized.</li> <li>• +1: pure perpendicular (s) polarization.</li> </ul>
----------------------------	--

## Returns

const reference to layer layer\_no

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

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

- rl\_ReflectionCoefPOD.h

## 3.13 rl\_Traits Class Reference

[rl\\_Traits](#) is a "traits" class for the [rl\\_RayLib](#) library.

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

## Classes

- struct [rl\\_DielectricPOD](#)

## Public Types

- enum [Bool](#)  
*Typedef for the Boolean type.*
- enum [EInterpMode](#) { [ELinLin](#), [ELinLog](#), [ELogLin](#), [ELogLog](#) }  
*Enumeration specifying the interpolation of the optical constants.*
- enum [ERoughType](#) {  
  [ERoughNone](#), [ERoughDebyeWaller\\_RSABO](#), [ERoughDebyeWaller\\_CSABO](#), [ERoughDebyeWaller\\_Spiller](#),  
  [ERoughModifiedDebyeWaller](#), [ERoughNevotCroce](#) }  
*Enumeration specifying the type of interlayer diffusion treatment.*
- typedef std::complex< double > [complex](#)  
*Typedef for the complex type.*

### 3.13.1 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](#).

### 3.13.2 Member Enumeration Documentation

#### 3.13.2.1 EInterpMode

```
enum rl\_Traits::EInterpMode
```

Enumeration specifying the interpolation of the optical constants.

#### Enumerator

<a href="#">ELinLin</a>	linear in energy, linear in optical constants.
<a href="#">ELinLog</a>	log in energy, linear in optical constants.
<a href="#">ELogLin</a>	linear in energy, log in optical constants.
<a href="#">ELogLog</a>	log in energy, log in optical constants.

Definition at line 69 of file [rl\\_Traits.h](#).

## 3.13.2.2 ERoughType

```
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

## 3.14 rl\_TransmissionCoefPOD Class Reference

A Plain Ol' Data class representing complex reflection coefficients.

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

## Public Member Functions

- void [init](#) ()  
*Initialize perpendicular (s) and parallel (p) transmission coefficients to zero.*
- void [init](#) (rl\_Traits::complex &para, rl\_Traits::complex &perp)  
*Initialize perpendicular (s) and parallel (p) transmission coefficients.*
- double [transmission](#) (double polarization\_factor=0.0) const  
*Transmission factor.*
- 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  
*Print reflectivity information to output stream.*
- void [cprint\\_on](#) (std::FILE \*of, char const pre[]="", char const pst[]="") const  
*Print reflectivity information to output FILE\* stream.*

### 3.14.1 Detailed Description

A Plain Ol' Data class representing complex reflection coefficients.

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

### 3.14.2 Member Function Documentation

#### 3.14.2.1 cprint\_on()

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

Print reflectivity information to output FILE\* stream.

##### Parameters

<i>of</i>	output FILE* stream.
<i>pre</i>	optional prefix (char*) string.
<i>pst</i>	optional postfix (char*) string.

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

#### 3.14.2.2 init()

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

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

##### Parameters

<i>para</i>	parallel transmission coefficient
<i>perp</i>	perpendicular transmission coefficient

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



#### 3.14.2.3 para() [1/2]

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

##### Returns

parallel (p) transmission coefficient.

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

#### 3.14.2.4 para() [2/2]

```
rl_Traits::complex & rl_TransmissionCoefPOD::para ( ) [inline]
```

##### Returns

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

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

#### 3.14.2.5 perp() [1/2]

```
rl_Traits::complex rl_TransmissionCoefPOD::perp ( ) const [inline]
```

##### Returns

perpendicular (s) transmission coefficient.

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

#### 3.14.2.6 perp() [2/2]

```
rl_Traits::complex & rl_TransmissionCoefPOD::perp ( ) [inline]
```

##### Returns

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

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

#### 3.14.2.7 print\_on()

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

Print reflectivity information to output stream.

## Parameters

<i>os</i>	output stream.
<i>pre</i>	optional prefix (char*) string.
<i>pst</i>	optional postfix (char*) string.

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

## 3.14.2.8 transmission()

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

Transmission factor.

## Parameters

<i>polarization_factor</i>	<p>polarization factor; it must be a value between -1 and 1. The polarization factor is related to parallel (p) and perpendicular (s) polarization by:</p> $\text{polarization\_factor} = \frac{(I_{\perp} - I_{\parallel})}{(I_{\perp} + I_{\parallel})}$ <p>or</p> $\text{polarization\_factor} = \frac{(I_s - I_p)}{(I_s + I_p)}$ <p>where <math>I_{\perp}</math> and <math>I_{\parallel}</math> are the perpendicular and parallel E-field <i>intensities</i>, respectively. Thus,</p> <ul style="list-style-type: none"> <li>• -1: pure parallel (p) polarization.</li> <li>• 0: completely unpolarized.</li> <li>• +1: pure perpendicular (s) polarization.</li> </ul>
----------------------------	--

## 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\_Multilayer
  - rl\_Multilayer, [27](#)
- alpha
  - rl\_DielectricLayer, [12](#)
- alpha\_gamma
  - rl\_DielectricData, [7](#)
- attenuate
  - rl\_Polarization, [37](#)
  - rl\_PolCSPOD, [42](#)
  - rl\_Ray, [46](#)
- bulk\_density\_factor
  - rl\_DielectricData, [7](#)
  - rl\_DielectricLayer, [12](#)
- cdump\_on
  - rl\_DielectricLayer, [13](#)
  - rl\_Multilayer, [28](#)
- const\_data\_ptr
  - rl\_DielectricPODArray, [23](#)
- cprint\_constraints\_on
  - rl\_DielectricLayer, [13](#)
- cprint\_on
  - rl\_DielectricPODArray, [23](#)
  - rl\_Polarization, [37](#)
  - rl\_PolCSPOD, [42](#)
  - rl\_ReflectionCoefPOD, [50](#)
  - rl\_TransmissionCoefPOD, [56](#)
- derotate
  - rl\_Polarization, [38](#)
- derotate\_detranslate
  - rl\_Ray, [46](#)
- dump\_on
  - rl\_DielectricLayer, [14](#)
  - rl\_Multilayer, [28](#)
  - rl\_MultilayerSurface, [33](#)
- EInterpMode
  - rl\_Traits, [54](#)
- ELinLin
  - rl\_Traits, [54](#)
- ELinLog
  - rl\_Traits, [54](#)
- ELogLin
  - rl\_Traits, [54](#)
- ELogLog
  - rl\_Traits, [54](#)
- energy\_max
  - rl\_DielectricLayer, [14](#)
- energy\_min
  - rl\_DielectricLayer, [14](#)
- ERoughDebyeWaller\_CSAO
  - rl\_Traits, [55](#)
- ERoughDebyeWaller\_RSAO
  - rl\_Traits, [55](#)
- ERoughDebyeWaller\_Spiller
  - rl\_Traits, [55](#)
- ERoughModifiedDebyeWaller
  - rl\_Traits, [55](#)
- ERoughNevotCroce
  - rl\_Traits, [55](#)
- ERoughNone
  - rl\_Traits, [55](#)
- ERoughType
  - rl\_Traits, [54](#)
- gamma
  - rl\_DielectricLayer, [14](#)
- get\_PolCSPOD
  - rl\_Polarization, [38](#)
  - rl\_Ray, [46](#)
- init
  - rl\_DielectricData, [8](#)
  - rl\_DielectricLayer, [15](#)
  - rl\_DielectricPODArray, [23](#), [24](#)
  - rl\_Multilayer, [29](#)
  - rl\_Polarization, [39](#)
  - rl\_PolCSPOD, [42](#)
  - rl\_ReflectionCoefPOD, [50](#)
  - rl\_TransmissionCoefPOD, [56](#)
- init\_ray
  - rl\_Ray, [47](#)
- intensity
  - rl\_Polarization, [39](#)
  - rl\_PolCSPOD, [43](#)
  - rl\_Ray, [47](#)
- is\_substrate
  - rl\_DielectricLayer, [15](#)
- is\_vacuum

- rl\_DielectricLayer, 16
- layer
  - rl\_Multilayer, 29
  - rl\_MultilayerSurface, 33
- layer\_name
  - rl\_DielectricLayer, 16
- multilayer\_reflect\_coef
  - rl\_Multilayer, 29
- multilayer\_reflectivity
  - rl\_Multilayer, 30
- normal\_vector
  - rl\_MultilayerSurface, 33
- num\_elts
  - rl\_DielectricPODArray, 24
- num\_layers
  - rl\_Multilayer, 31
- para
  - rl\_ReflectionCoefPOD, 51
  - rl\_TransmissionCoefPOD, 56, 57
- perp
  - rl\_ReflectionCoefPOD, 51, 52
  - rl\_TransmissionCoefPOD, 57
- polarization
  - rl\_Ray, 47
- print\_on
  - rl\_Polarization, 39
  - rl\_PolCSPOD, 43
  - rl\_Ray, 48
  - rl\_ReflectionCoefPOD, 52
  - rl\_TransmissionCoefPOD, 57
- propagator
  - rl\_DielectricLayer, 16
- reflect
  - rl\_MultilayerSurface, 34
  - rl\_Polarization, 40
  - rl\_Ray, 48
- reflect\_amp
  - rl\_DielectricLayer, 16
- reflect\_nlayer
  - rl\_DielectricLayer, 17
- reflection\_coef
  - rl\_DielectricLayer, 17
- reflection\_coefs
  - rl\_MultilayerSurface, 34
- reflectivity
  - rl\_DielectricLayer, 17
  - rl\_ReflectionCoefPOD, 52
- rl\_DielectricData, 5
  - alpha\_gamma, 7
  - bulk\_density\_factor, 7
  - init, 8
  - rl\_DielectricData, 6
- rl\_DielectricData::rl\_DielectricDataBinPOD, 8
- rl\_DielectricLayer, 9
  - alpha, 12
  - bulk\_density\_factor, 12
  - cdump\_on, 13
  - cprint\_constraints\_on, 13
  - dump\_on, 14
  - energy\_max, 14
  - energy\_min, 14
  - gamma, 14
  - init, 15
  - is\_substrate, 15
  - is\_vacuum, 16
  - layer\_name, 16
  - propagator, 16
  - reflect\_amp, 16
  - reflect\_nlayer, 17
  - reflection\_coef, 17
  - reflectivity, 17
  - rl\_DielectricLayer, 11
  - roughness, 18
  - roughness\_type, 18
  - setup\_for, 18
  - thickness, 19
  - zcoat, 19
- rl\_DielectricPODArray, 20
  - const\_data\_ptr, 23
  - cprint\_on, 23
  - init, 23, 24
  - num\_elts, 24
  - rl\_DielectricPODArray, 21, 22
- rl\_Exception, 25
- rl\_Multilayer, 26
  - ~rl\_Multilayer, 27
  - cdump\_on, 28
  - dump\_on, 28
  - init, 29
  - layer, 29
  - multilayer\_reflect\_coef, 29
  - multilayer\_reflectivity, 30
  - num\_layers, 31
  - rl\_Multilayer, 27
- rl\_MultilayerSurface, 31
  - dump\_on, 33
  - layer, 33
  - normal\_vector, 33
  - reflect, 34
  - reflection\_coefs, 34
  - rl\_MultilayerSurface, 32
  - set\_normal, 34
- rl\_Polarization, 35
  - attenuate, 37

- cprint\_on, 37
- derotate, 38
- get\_PolCSPOD, 38
- init, 39
- intensity, 39
- print\_on, 39
- reflect, 40
- rl\_Polarization, 36, 37
- rotate, 40
- rl\_PolCSPOD, 41
  - attenuate, 42
  - cprint\_on, 42
  - init, 42
  - intensity, 43
  - print\_on, 43
- rl\_Ray, 44
  - attenuate, 46
  - derotate\_detranslate, 46
  - get\_PolCSPOD, 46
  - init\_ray, 47
  - intensity, 47
  - polarization, 47
  - print\_on, 48
  - reflect, 48
  - rl\_Ray, 45
  - set\_polarization, 49
  - translate\_rotate, 49
- rl\_ReflectionCoefPOD, 49
  - cprint\_on, 50
  - init, 50
  - para, 51
  - perp, 51, 52
  - print\_on, 52
  - reflectivity, 52
- rl\_Traits, 53
  - EInterpMode, 54
  - ELinLin, 54
  - ELinLog, 54
  - ELogLin, 54
  - ELogLog, 54
  - ERoughDebyeWaller\_CSAO, 55
  - ERoughDebyeWaller\_RSAO, 55
  - ERoughDebyeWaller\_Spiller, 55
  - ERoughModifiedDebyeWaller, 55
  - ERoughNevotCroce, 55
  - ERoughNone, 55
  - ERoughType, 54
- rl\_Traits::rl\_DielectricPOD, 19
- rl\_TransmissionCoefPOD, 55
  - cprint\_on, 56
  - init, 56
  - para, 56, 57
  - perp, 57
  - print\_on, 57
  - transmission, 58
- rotate
  - rl\_Polarization, 40
- roughness
  - rl\_DielectricLayer, 18
- roughness\_type
  - rl\_DielectricLayer, 18
- set\_normal
  - rl\_MultilayerSurface, 34
- set\_polarization
  - rl\_Ray, 49
- setup\_for
  - rl\_DielectricLayer, 18
- thickness
  - rl\_DielectricLayer, 19
- translate\_rotate
  - rl\_Ray, 49
- transmission
  - rl\_TransmissionCoefPOD, 58
- zcoat
  - rl\_DielectricLayer, 19