

Online Histogramming Reference Manual

Core

Version 0.01

Contents

1	OH Core	1
1.1	Extending the OH	1
2	OH Core Hierarchical Index	3
2.1	OH Core Class Hierarchy	3
3	OH Core Compound Index	5
3.1	OH Core Compound List	5
4	OH Core Page Index	7
4.1	OH Core Related Pages	7
5	OH Core Class Documentation	9
5.1	ISHistogram1D Class Reference	9
5.2	ISHistogram2D Class Reference	14
5.3	ISHistogram3D Class Reference	19
5.4	ISHistogramSet1D Class Template Reference	25
5.5	ISHistogramSet2D Class Template Reference	27
5.6	ISHistogramSet3D Class Template Reference	29
5.7	OHHistogramData Class Reference	31
5.8	OHHistogramData1D Class Reference	38
5.9	OHHistogramData2D Class Reference	48
5.10	OHHistogramData3D Class Reference	59
5.11	OHHistogramIterator Class Reference	70
5.12	OHHistogramProvider Class Reference	77
5.13	OHHistogramReceiver Class Reference	85
5.14	OHHistogramSubscriber Class Reference	88
5.15	OHProviderIterator Class Reference	93
5.16	OHRawProvider Class Template Reference	97

5.17 OHRawProviderDM Class Reference	106
5.18 OHRootProvider Class Reference	108
5.19 OHRootReceiver Class Reference	112
5.20 OHServerIterator Class Reference	115
6 OH Core Page Documentation	119
6.1 Naming conventions	119
6.2 The ROOT Histogram Provider	120
6.3 The ROOT Histogram Receiver	121
6.4 Todo List	122

Chapter 1

OH Core

This manual contains information about the OH core in addition to what is available in the public manual, some sourcecode to the OH is also included.

The core contains parts of the OH which should not be used directly by histogram provider tasks or user histogram tasks and this documentation is primarily for maintenance purpose or people who intend to extend the OH by adding histogram import or export functionality.

1.1 Extending the OH

The OH core is designed to provide a generic interface for histogram representation, selection and transportation. The core provides a foundation for importing and exporting histograms, the actual user interfaces are provided by specific provider and receiver classes in the public package.

1.1.1 Adding a provider class to the OH

Check the documentation for **OHHistogramProvider** (p. 77) to find out how to add a new provider class. Also check the implementation of the currently existing provider classes

- **OHRawProvider** (p. 97)
- **OHRootProvider** (p. 108)

1.1.2 Adding a receiver class to the OH

Check the documentation for **OHHistogramReceiver** (p. 85) to find out how to add a new receiver class. Also check the implementation of the currently existing receiver classes

- **OHRootReceiver** (p. 112)
-

Chapter 2

OH Core Hierarchical Index

2.1 OH Core Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ISHistogram1D	9
OHHistogramData1D	38
ISHistogram2D	14
OHHistogramData2D	48
ISHistogram3D	19
OHHistogramData3D	59
ISHistogramSet1D< T >	25
ISHistogramSet2D< T >	27
ISHistogramSet3D< T >	29
OHHistogramData	31
OHHistogramData1D	38
OHHistogramData2D	48
OHHistogramData3D	59
OHHistogramIterator	70
OHHistogramProvider	77
OHRawProvider< TContent, TError, TAxis, TMap >	97
OHRootProvider	108
OHHistogramReceiver	85
OHRootReceiver	112
OHHistogramSubscriber	88
OHProviderIterator	93
OHRawProviderDM	106
OHServerIterator	115
OHUtils	

Chapter 3

OH Core Compound Index

3.1 OH Core Compound List

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

ISHistogram1D	9
ISHistogram2D	14
ISHistogram3D	19
ISHistogramSet1D < T > (A set of one dimensional histograms)	25
ISHistogramSet2D < T > (A set of two dimensional histograms)	27
ISHistogramSet3D < T > (A set of three dimensional histograms)	29
OHHistogramData (Histogram data interface)	31
OHHistogramData1D (One dimensional histogram data)	38
OHHistogramData2D (Two dimensional histogram data)	48
OHHistogramData3D (Three dimensional histogram data)	59
OHHistogramIterator (Histogram Iterator)	70
OHHistogramProvider (Histogram Provider baseclass)	77
OHHistogramReceiver (Histogram Receiver baseclass)	85
OHHistogramSubscriber (Histogram subscriber)	88
OHProviderIterator (Provider iterator)	93
OHRawProvider < TContent , TError , TAxis , TMap > (RAW Provider)	97
OHRawProviderDM (RAW Provider default map)	106
OHRootProvider (ROOT Provider)	108
OHRootReceiver (ROOT Receiver)	112
OHServerIterator (Server iterator)	115

Chapter 4

OH Core Page Index

4.1 OH Core Related Pages

Here is a list of all related documentation pages:

Naming conventions	119
The ROOT Histogram Provider	120
The ROOT Histogram Receiver	121
Todo List	122

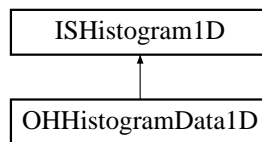
Chapter 5

OH Core Class Documentation

5.1 ISHistogram1D Class Reference

```
#include <ISHistogram1D.h>
```

Inheritance diagram for ISHistogram1D::



Public Types

- enum `is_xtype_E` { `IS_FIXED`, `IS_VARIABLE` }

Public Methods

- `ISHistogram1D ()`
- `~ISHistogram1D ()`

Public Attributes

- string `is_title`
 - string * `is_annotations`
 - size_t `is_annotations_size`
 - `is_xtype_E` `is_xtype`
 - string `is_xlabel`
 - double * `is_xaxis`
 - size_t `is_xaxis_size`
 - double * `is_bins`
 - size_t `is_bins_size`
 - double * `is_errors`
 - size_t `is_errors_size`
-

- double * **is_plusererrors**
- size_t **is_plusererrors_size**
- double * **is_minusererrors**
- size_t **is_minusererrors_size**
- double **is_xmean**
- double **is_xrms**

Protected Methods

- **ISHistogram1D** (const char *type)
- void **publishGuts** (ISostream &out)
- void **refreshGuts** (ISistream &in)

5.1.1 Detailed Description

One dimensional histogram

Author:

generated by the IS tool

Version:

08/04/02

Definition at line 16 of file ISHistogram1D.h.

5.1.2 Member Data Documentation

5.1.2.1 string* ISHistogram1D::is_annotiations

Annotations

Definition at line 28 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_annotiations, and OHHistogramData1D::set_annotiations.

5.1.2.2 size_t ISHistogram1D::is_annotiations_size

size of the is_annotiations array

Definition at line 32 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_annotiations, and OHHistogramData1D::set_annotiations.

5.1.2.3 double* ISHistogram1D::is_bins

Bin values

Definition at line 56 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_height, OHHistogramData1D::reset_bins, and OHHistogramData1D::set_height.

5.1.2.4 `size_t` ISHistogram1D::is_bins_size

size of the is_bins array

Definition at line 60 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_height, and OHHistogramData1D::reset_bins.

5.1.2.5 `double*` ISHistogram1D::is_errors

Optional error values for the bins

Definition at line 65 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_error, OHHistogramData1D::get_perror, OHHistogramData1D::reset_bins, and OHHistogramData1D::set_error.

5.1.2.6 `size_t` ISHistogram1D::is_errors_size

size of the is_errors array

Definition at line 69 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::error_type, OHHistogramData1D::get_error, OHHistogramData1D::get_perror, OHHistogramData1D::reset_bins, and OHHistogramData1D::set_perror.

5.1.2.7 `double*` ISHistogram1D::is_minuserrors

Optional minuserror values for the bins

Definition at line 83 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_error, OHHistogramData1D::get_perror, OHHistogramData1D::reset_bins, and OHHistogramData1D::set_perror.

5.1.2.8 `size_t` ISHistogram1D::is_minuserrors_size

size of the is_minuserrors array

Definition at line 87 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::error_type, OHHistogramData1D::get_error, OHHistogramData1D::get_perror, OHHistogramData1D::reset_bins, and OHHistogramData1D::set_error.

5.1.2.9 `double*` ISHistogram1D::is_pluserrors

Optional plusererror values for the bins

Definition at line 74 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_error, OHHistogramData1D::get_perror, OHHistogramData1D::reset_bins, and OHHistogramData1D::set_perror.

5.1.2.10 size_t ISHistogram1D::is_pluserrors_size

size of the is_pluserrors array

Definition at line 78 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::error_type, OHHistogramData1D::get_error, OHHistogramData1D::get_pmerror, OHHistogramData1D::reset_bins, and OHHistogramData1D::set_error.

5.1.2.11 string ISHistogram1D::is_title

Histogram title

Definition at line 23 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_title, and OHHistogramData1D::set_title.

5.1.2.12 double* ISHistogram1D::is_xaxis

X-Axis partition, content depends on xtype

Definition at line 47 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::bin_count, OHHistogramData1D::get_partition, and OHHistogramData1D::set_partition.

5.1.2.13 size_t ISHistogram1D::is_xaxis_size

size of the is_xaxis array

Definition at line 51 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::bin_count, OHHistogramData1D::get_partition, and OHHistogramData1D::set_partition.

5.1.2.14 string ISHistogram1D::is_xlabel

X-Axis label

Definition at line 42 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_label, and OHHistogramData1D::set_label.

5.1.2.15 double ISHistogram1D::is_xmean

Mean of the whole histogram

Definition at line 92 of file ISHistogram1D.h.

Referenced by OHHistogramData1D::get_mean, and OHHistogramData1D::set_mean.

5.1.2.16 double ISHistogram1D::is_xrms

RMS of the whole histogram

Definition at line 97 of file ISHistogram1D.h.

Referenced by `OHHistogramData1D::get_rms`, and `OHHistogramData1D::set_rms`.

5.1.2.17 `is_xtype_E` `ISHistogram1D::is_xtype`

X-Axis type

Definition at line 37 of file `ISHistogram1D.h`.

Referenced by `OHHistogramData1D::axis_type`, `OHHistogramData1D::bin_count`, `OHHistogramData1D::get_partition`, and `OHHistogramData1D::set_partition`.

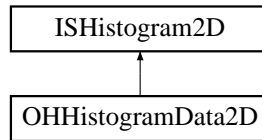
The documentation for this class was generated from the following file:

- `ISHistogram1D.h`

5.2 ISHistogram2D Class Reference

```
#include <ISHistogram2D.h>
```

Inheritance diagram for ISHistogram2D::



Public Types

- enum `is_xtype_E` { `IS_FIXED`, `IS_VARIABLE` }

Public Methods

- `ISHistogram2D ()`
- `~ISHistogram2D ()`

Public Attributes

- string `is_title`
- string * `is_annotations`
- `size_t` `is_annotations_size`
- `is_xtype_E` `is_xtype`
- string `is_xlabel`
- double * `is_xaxis`
- `size_t` `is_xaxis_size`
- `is_xtype_E` `is_ytype`
- string `is_ylabel`
- double * `is_yaxis`
- `size_t` `is_yaxis_size`
- double * `is_bins`
- `size_t` `is_bins_size`
- double * `is_errors`
- `size_t` `is_errors_size`
- double * `is_pluserrors`
- `size_t` `is_pluserrors_size`
- double * `is_minusererrors`
- `size_t` `is_minusererrors_size`
- double `is_xmean`
- double `is_xrms`
- double `is_ymean`
- double `is_yrms`

Protected Methods

- **ISHistogram2D** (const char *type)
- void **publishGuts** (ISostream &out)
- void **refreshGuts** (ISistream &in)

5.2.1 Detailed Description

Two dimensional histogram

Author:

generated by the IS tool

Version:

08/04/02

Definition at line 16 of file ISHistogram2D.h.

5.2.2 Member Data Documentation

5.2.2.1 string* ISHistogram2D::is_annotatations

Annotations

Definition at line 28 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_annotatations, and OHHistogramData2D::set_annotatations.

5.2.2.2 size_t ISHistogram2D::is_annotatations_size

size of the is_annotatations array

Definition at line 32 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_annotatations, and OHHistogramData2D::set_annotatations.

5.2.2.3 double* ISHistogram2D::is_bins

Bin values

Definition at line 75 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_height, OHHistogramData2D::reset_bins, and OHHistogramData2D::set_height.

5.2.2.4 size_t ISHistogram2D::is_bins_size

size of the is_bins array

Definition at line 79 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_height, and OHHistogramData2D::reset_bins.

5.2.2.5 `double*` `ISHistogram2D::is_errors`

Optional error values for the bins

Definition at line 84 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::get_error`, `OHHistogramData2D::get_pmerror`, `OHHistogramData2D::reset_bins`, and `OHHistogramData2D::set_error`.

5.2.2.6 `size_t` `ISHistogram2D::is_errors_size`

size of the `is_errors` array

Definition at line 88 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::error_type`, `OHHistogramData2D::get_error`, `OHHistogramData2D::get_pmerror`, `OHHistogramData2D::reset_bins`, and `OHHistogramData2D::set_pmerror`.

5.2.2.7 `double*` `ISHistogram2D::is_minuserrors`

Optional minusererror values for the bins

Definition at line 102 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::get_error`, `OHHistogramData2D::get_pmerror`, `OHHistogramData2D::reset_bins`, and `OHHistogramData2D::set_pmerror`.

5.2.2.8 `size_t` `ISHistogram2D::is_minuserrors_size`

size of the `is_minuserrors` array

Definition at line 106 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::error_type`, `OHHistogramData2D::get_error`, `OHHistogramData2D::get_pmerror`, `OHHistogramData2D::reset_bins`, and `OHHistogramData2D::set_error`.

5.2.2.9 `double*` `ISHistogram2D::is_plusererrors`

Optional plusererror values for the bins

Definition at line 93 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::get_error`, `OHHistogramData2D::get_pmerror`, `OHHistogramData2D::reset_bins`, and `OHHistogramData2D::set_pmerror`.

5.2.2.10 `size_t` `ISHistogram2D::is_plusererrors_size`

size of the `is_plusererrors` array

Definition at line 97 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::error_type`, `OHHistogramData2D::get_error`, `OHHistogramData2D::get_pmerror`, `OHHistogramData2D::reset_bins`, and `OHHistogramData2D::set_error`.

5.2.2.11 string ISHistogram2D::is_title

Histogram title

Definition at line 23 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_title, and OHHistogramData2D::set_title.

5.2.2.12 double* ISHistogram2D::is_xaxis

X-Axis partition, content depends on xtype

Definition at line 47 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::bin_count, OHHistogramData2D::get_partition, and OHHistogramData2D::set_partition.

5.2.2.13 size_t ISHistogram2D::is_xaxis_size

size of the is_xaxis array

Definition at line 51 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::bin_count, OHHistogramData2D::get_partition, and OHHistogramData2D::set_partition.

5.2.2.14 string ISHistogram2D::is_xlabel

X-Axis label

Definition at line 42 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_label, and OHHistogramData2D::set_label.

5.2.2.15 double ISHistogram2D::is_xmean

Mean of the whole histogram projected to the x-axis

Definition at line 111 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_mean, and OHHistogramData2D::set_mean.

5.2.2.16 double ISHistogram2D::is_xrms

RMS of the whole histogram projected to the x-axis

Definition at line 116 of file ISHistogram2D.h.

Referenced by OHHistogramData2D::get_rms, and OHHistogramData2D::set_rms.

5.2.2.17 is_xtype_E ISHistogram2D::is_xtype

X-Axis type

Definition at line 37 of file ISHistogram2D.h.

Referenced by `OHHistogramData2D::axis_type`, `OHHistogramData2D::bin_count`, `OHHistogramData2D::get_partition`, and `OHHistogramData2D::set_partition`.

5.2.2.18 `double*` `ISHistogram2D::is_yaxis`

Y-Axis partition, content depends on `ytype`

Definition at line 66 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::bin_count`, `OHHistogramData2D::get_partition`, and `OHHistogramData2D::set_partition`.

5.2.2.19 `size_t` `ISHistogram2D::is_yaxis_size`

size of the `is_yaxis` array

Definition at line 70 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::bin_count`, `OHHistogramData2D::get_partition`, and `OHHistogramData2D::set_partition`.

5.2.2.20 `string` `ISHistogram2D::is_ylabel`

Y-Axis label

Definition at line 61 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::get_label`, and `OHHistogramData2D::set_label`.

5.2.2.21 `double` `ISHistogram2D::is_ymean`

Mean of the whole histogram projected to the y-axis

Definition at line 121 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::get_mean`, and `OHHistogramData2D::set_mean`.

5.2.2.22 `double` `ISHistogram2D::is_yrms`

RMS of the whole histogram projected to the y-axis

Definition at line 126 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::get_rms`, and `OHHistogramData2D::set_rms`.

5.2.2.23 `is_xtype_E` `ISHistogram2D::is_ytype`

Y-Axis type

Definition at line 56 of file `ISHistogram2D.h`.

Referenced by `OHHistogramData2D::axis_type`, `OHHistogramData2D::bin_count`, `OHHistogramData2D::get_partition`, and `OHHistogramData2D::set_partition`.

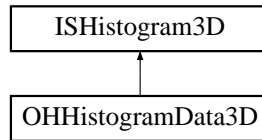
The documentation for this class was generated from the following file:

- `ISHistogram2D.h`

5.3 ISHistogram3D Class Reference

```
#include <ISHistogram3D.h>
```

Inheritance diagram for ISHistogram3D::



Public Types

- enum `is_xtype_E` { `IS_FIXED`, `IS_VARIABLE` }

Public Methods

- `ISHistogram3D ()`
- `~ISHistogram3D ()`

Public Attributes

- string `is_title`
- string * `is_annotations`
- size_t `is_annotations_size`
- `is_xtype_E` `is_xtype`
- string `is_xlabel`
- double * `is_xaxis`
- size_t `is_xaxis_size`
- `is_xtype_E` `is_ytype`
- string `is_ylabel`
- double * `is_yaxis`
- size_t `is_yaxis_size`
- `is_xtype_E` `is_ztype`
- string `is_zlabel`
- double * `is_zaxis`
- size_t `is_zaxis_size`
- double * `is_bins`
- size_t `is_bins_size`
- double * `is_errors`
- size_t `is_errors_size`
- double * `is_pluserrors`
- size_t `is_pluserrors_size`
- double * `is_minusererrors`
- size_t `is_minusererrors_size`
- double `is_xmean`
- double `is_xrms`
- double `is_ymean`

- double **is_yrms**
- double **is_zmean**
- double **is_zrms**

Protected Methods

- **ISHistogram3D** (const char *type)
- void **publishGuts** (ISostream &out)
- void **refreshGuts** (ISistream &in)

5.3.1 Detailed Description

Three dimensional histogram

Author:

generated by the IS tool

Version:

08/04/02

Definition at line 16 of file ISHistogram3D.h.

5.3.2 Member Data Documentation

5.3.2.1 string* ISHistogram3D::is_annotations

Annotations

Definition at line 28 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_annotations`, and `OHHistogramData3D::set_annotations`.

5.3.2.2 size_t ISHistogram3D::is_annotations_size

size of the is_annotations array

Definition at line 32 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_annotations`, and `OHHistogramData3D::set_annotations`.

5.3.2.3 double* ISHistogram3D::is_bins

Bin values

Definition at line 94 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_height`, `OHHistogramData3D::reset_bins`, and `OHHistogramData3D::set_height`.

5.3.2.4 `size_t` ISHistogram3D::is_bins_size

size of the `is_bins` array

Definition at line 98 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_height`, and `OHHistogramData3D::reset_bins`.

5.3.2.5 `double*` ISHistogram3D::is_errors

Optional error values for the bins

Definition at line 103 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_error`, `OHHistogramData3D::get_perror`, `OHHistogramData3D::reset_bins`, and `OHHistogramData3D::set_error`.

5.3.2.6 `size_t` ISHistogram3D::is_errors_size

size of the `is_errors` array

Definition at line 107 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::error_type`, `OHHistogramData3D::get_error`, `OHHistogramData3D::get_perror`, `OHHistogramData3D::reset_bins`, and `OHHistogramData3D::set_perror`.

5.3.2.7 `double*` ISHistogram3D::is_minuserrors

Optional minusererror values for the bins

Definition at line 121 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_error`, `OHHistogramData3D::get_perror`, `OHHistogramData3D::reset_bins`, and `OHHistogramData3D::set_perror`.

5.3.2.8 `size_t` ISHistogram3D::is_minuserrors_size

size of the `is_minuserrors` array

Definition at line 125 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::error_type`, `OHHistogramData3D::get_error`, `OHHistogramData3D::get_perror`, `OHHistogramData3D::reset_bins`, and `OHHistogramData3D::set_error`.

5.3.2.9 `double*` ISHistogram3D::is_plusererrors

Optional plusererror values for the bins

Definition at line 112 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_error`, `OHHistogramData3D::get_perror`, `OHHistogramData3D::reset_bins`, and `OHHistogramData3D::set_perror`.

5.3.2.10 size_t ISHistogram3D::is_plusererrors_size

size of the is_plusererrors array

Definition at line 116 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::error_type, OHHistogramData3D::get_error, OHHistogramData3D::get_pmerror, OHHistogramData3D::reset_bins, and OHHistogramData3D::set_error.

5.3.2.11 string ISHistogram3D::is_title

Histogram title

Definition at line 23 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::get_title, and OHHistogramData3D::set_title.

5.3.2.12 double* ISHistogram3D::is_xaxis

X-Axis partition, content depends on xtype

Definition at line 47 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::bin_count, OHHistogramData3D::get_partition, and OHHistogramData3D::set_partition.

5.3.2.13 size_t ISHistogram3D::is_xaxis_size

size of the is_xaxis array

Definition at line 51 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::bin_count, OHHistogramData3D::get_partition, and OHHistogramData3D::set_partition.

5.3.2.14 string ISHistogram3D::is_xlabel

X-Axis label

Definition at line 42 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::get_label, and OHHistogramData3D::set_label.

5.3.2.15 double ISHistogram3D::is_xmean

Mean of the whole histogram projected to the x-axis

Definition at line 130 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::get_mean, and OHHistogramData3D::set_mean.

5.3.2.16 double ISHistogram3D::is_xrms

RMS of the whole histogram projected to the x-axis

Definition at line 135 of file ISHistogram3D.h.

Referenced by `OHHistogramData3D::get_rms`, and `OHHistogramData3D::set_rms`.

5.3.2.17 `is_xtype_E` `ISHistogram3D::is_xtype`

X-Axis type

Definition at line 37 of file `ISHistogram3D.h`.

Referenced by `OHHistogramData3D::axis_type`, `OHHistogramData3D::bin_count`, `OHHistogramData3D::get_partition`, and `OHHistogramData3D::set_partition`.

5.3.2.18 `double*` `ISHistogram3D::is_yaxis`

Y-Axis partition, content depends on `ytype`

Definition at line 66 of file `ISHistogram3D.h`.

Referenced by `OHHistogramData3D::bin_count`, `OHHistogramData3D::get_partition`, and `OHHistogramData3D::set_partition`.

5.3.2.19 `size_t` `ISHistogram3D::is_yaxis_size`

size of the `is_yaxis` array

Definition at line 70 of file `ISHistogram3D.h`.

Referenced by `OHHistogramData3D::bin_count`, `OHHistogramData3D::get_partition`, and `OHHistogramData3D::set_partition`.

5.3.2.20 `string` `ISHistogram3D::is_ylabel`

Y-Axis label

Definition at line 61 of file `ISHistogram3D.h`.

Referenced by `OHHistogramData3D::get_label`, and `OHHistogramData3D::set_label`.

5.3.2.21 `double` `ISHistogram3D::is_ymean`

Mean of the whole histogram projected to the y-axis

Definition at line 140 of file `ISHistogram3D.h`.

Referenced by `OHHistogramData3D::get_mean`, and `OHHistogramData3D::set_mean`.

5.3.2.22 `double` `ISHistogram3D::is_yrms`

RMS of the whole histogram projected to the y-axis

Definition at line 145 of file `ISHistogram3D.h`.

Referenced by `OHHistogramData3D::get_rms`, and `OHHistogramData3D::set_rms`.

5.3.2.23 `is_xtype_E` `ISHistogram3D::is_ytype`

Y-Axis type

Definition at line 56 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::axis_type, OHHistogramData3D::bin_count, OHHistogramData3D::get_partition, and OHHistogramData3D::set_partition.

5.3.2.24 double* ISHistogram3D::is_zaxis

Z-Axis partition, content depends on ztype

Definition at line 85 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::bin_count, OHHistogramData3D::get_partition, and OHHistogramData3D::set_partition.

5.3.2.25 size_t ISHistogram3D::is_zaxis_size

size of the is_zaxis array

Definition at line 89 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::bin_count, OHHistogramData3D::get_partition, and OHHistogramData3D::set_partition.

5.3.2.26 string ISHistogram3D::is_zlabel

Z-Axis label

Definition at line 80 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::get_label, and OHHistogramData3D::set_label.

5.3.2.27 double ISHistogram3D::is_zmean

Mean of the whole histogram projected to the z-axis

Definition at line 150 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::get_mean, and OHHistogramData3D::set_mean.

5.3.2.28 double ISHistogram3D::is_zrms

RMS of the whole histogram projected to the z-axis

Definition at line 155 of file ISHistogram3D.h.

5.3.2.29 is_xtype_E ISHistogram3D::is_ztype

Z-Axis type

Definition at line 75 of file ISHistogram3D.h.

Referenced by OHHistogramData3D::axis_type, OHHistogramData3D::bin_count, OHHistogramData3D::get_partition, and OHHistogramData3D::set_partition.

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

- ISHistogram3D.h

5.4 ISHistogramSet1D Class Template Reference

A set of one dimensional histograms.

```
#include <ISHistogramSet1D.h>
```

Public Methods

- **ISHistogramSet1D** (vector< T * > &set, bool clear=true)
Construct a (possibly empty) set of 1D histograms.
- **~ISHistogramSet1D** ()
Destructor.
- void **publishGuts** (ISostream &out)
Publish the set to the IS.
- void **refreshGuts** (ISistream &in)
Retreive a set from the IS.

5.4.1 Detailed Description

```
template<class T> class ISHistogramSet1D< T >
```

A set of one dimensional histograms.

The set is defined as a template in order to support a set of histogram objects which is derived from the **ISHistogram1D** (p.9) class (which contains the actual histogram data to be serialized for each histogram in the set).

T shall be a class derived from **ISHistogram1D** (p.9), the publishGuts and refreshGuts methods of **ISHistogram1D** (p.9) may Not be redefined.

Todo:

Find a way to serialize sets with IS (ISInfoAny? ISInfo Arrays?)

Definition at line 27 of file ISHistogramSet1D.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 `template<class T> ISHistogramSet1D< T >::ISHistogramSet1D (vector< T * > & set, bool clear = true) [inline]`

Construct a (possibly empty) set of 1D histograms.

Parameters:

set A possibly empty vector of **ISHistogram1D** (p.9) derived types.

clear Determines if the histograms in the vector shall be deleted when the object is destroyed or not. If the value is false any histograms remaining in the vector after the object is destroyed will be valid objects which the user is responsible for deleting. If the value is

true the vector will have size 0 after the object is destroyed. The vector will have size 0 also when the set is empty.

Definition at line 40 of file ISHistogramSet1D.h.

```
40                                     :
41         ISInfo( "ISHistogramSet1D" ),
42         is_set( set ), is_clear( clear ) {
43     }
```

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

- **ISHistogramSet1D.h**

5.5 ISHistogramSet2D Class Template Reference

A set of two dimensional histograms.

```
#include <ISHistogramSet2D.h>
```

Public Methods

- **ISHistogramSet2D** (vector< T * > &set, bool clear=true)
Construct a (possibly empty) set of 2D histograms.
- **~ISHistogramSet2D** ()
Destructor.
- void **publishGuts** (ISostream &out)
Publish the set to the IS.
- void **refreshGuts** (ISistream &in)
Retreive a set from the IS.

5.5.1 Detailed Description

```
template<class T> class ISHistogramSet2D< T >
```

A set of two dimensional histograms.

The set is defined as a template in order to support a set of histogram objects which is derived from the **ISHistogram2D** (p. 14) class (which contains the actual histogram data to be serialized for each histogram in the set).

T shall be a class derived from **ISHistogram2D** (p. 14), the **publishGuts** and **refreshGuts** methods of **ISHistogram2D** (p. 14) may Not be redefined.

Todo:

Find a way to serialize sets with IS (ISInfoAny? ISInfo Arrays?)

Definition at line 27 of file ISHistogramSet2D.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 `template<class T> ISHistogramSet2D< T >::ISHistogramSet2D (vector< T * > & set, bool clear = true) [inline]`

Construct a (possibly empty) set of 2D histograms.

Parameters:

set A possibly empty vector of **ISHistogram2D** (p. 14) derived types.

clear Determines if the histograms in the vector shall be deleted when the object is destroyed or not. If the value is false any histograms remaining in the vector after the object is destroyed will be valid objects which the user is responsible for deleting. If the value is

true the vector will have size 0 after the object is destroyed. The vector will have size 0 also when the set is empty.

Definition at line 40 of file ISHistogramSet2D.h.

```
40                                     :  
41         ISInfo( "ISHistogramSet2D" ),  
42         is_set( set ), is_clear( clear ) {  
43     }
```

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

- **ISHistogramSet2D.h**

5.6 ISHistogramSet3D Class Template Reference

A set of three dimensional histograms.

```
#include <ISHistogramSet3D.h>
```

Public Methods

- **ISHistogramSet3D** (vector< T * > &set, bool clear=true)
Construct a (possibly empty) set of 3D histograms.
- **~ISHistogramSet3D** ()
Destructor.
- void **publishGuts** (ISostream &out)
Publish the set to the IS.
- void **refreshGuts** (ISistream &in)
Retreive a set from the IS.

5.6.1 Detailed Description

```
template<class T> class ISHistogramSet3D< T >
```

A set of three dimensional histograms.

The set is defined as a template in order to support a set of histogram objects which is derived from the **ISHistogram3D** (p. 19) class (which contains the actual histogram data to be serialized for each histogram in the set).

T shall be a class derived from **ISHistogram3D** (p. 19), the publishGuts and refreshGuts methods of **ISHistogram3D** (p. 19) may Not be redefined.

Todo:

Find a way to serialize sets with IS (ISInfoAny? ISInfo Arrays?)

Definition at line 27 of file ISHistogramSet3D.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 `template<class T> ISHistogramSet3D< T >::ISHistogramSet3D (vector< T * > & set, bool clear = true) [inline]`

Construct a (possibly empty) set of 3D histograms.

Parameters:

set A possibly empty vector of **ISHistogram3D** (p. 19) derived types.

clear Determines if the histograms in the vector shall be deleted when the object is destroyed or not. If the value is false any histograms remaining in the vector after the object is destroyed will be valid objects which the user is responsible for deleting. If the value is

true the vector will have size 0 after the object is destroyed. The vector will have size 0 also when the set is empty.

Definition at line 40 of file ISHistogramSet3D.h.

```
40                                     :
41         ISInfo( "ISHistogramSet3D" ),
42         is_set( set ), is_clear( clear ) {
43     }
```

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

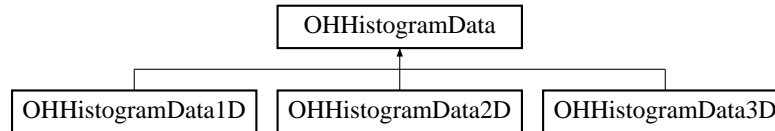
- **ISHistogramSet3D.h**

5.7 OHHistogramData Class Reference

Histogram data interface.

```
#include <OHHistogramData.h>
```

Inheritance diagram for OHHistogramData::



Public Types

- enum **Axis** { **X**, **Y**, **Z** }
Enumeration of the different axes.
- enum **AxisType** { **NoAxis**, **FixedAxis**, **VariableAxis** }
Enumeration of the different axis types.
- enum **ErrorType** { **NoErrors**, **SingleErrors**, **PlusMinusErrors** }
Enumeration of the different bin error types.

Public Methods

- virtual `~OHHistogramData ()`
Default destructor.
- virtual string `get_title (void) const=0`
Retrieve the title.
- virtual void `set_title (const string &title)=0`
Update the title.
- virtual long `get_annotations (vector< string > &labels, vector< string > &values) const=0`
Retrieve annotations.
- virtual void `set_annotations (const vector< string > &labels, const vector< string > &values)=0`
Update annotations.
- virtual string `get_label (Axis axis) const=0`
Retrieve label of a valid axis.
- virtual void `set_label (const string &label, Axis axis)=0`
Update label of a valid axis.

- virtual **AxisType** **axis_type** (**Axis** axis) const=0
Retrieve type of a valid axis.
- virtual long **bin_count** (**Axis** axis) const=0
Retrieve the number of bins for a valid axis.
- virtual void **get_partition** (vector< double > &partition, **Axis** axis) const=0
Retrieve a fixed or variable Patton for a valid axis.
- virtual void **get_partition** (double &offset, double &binwidth, long &bins, **Axis** axis) const=0
Retrieve a fixed Patton for a valid axis.
- virtual void **set_partition** (const vector< double > &partition, **Axis** axis)=0
Create a variable Patton for a valid axis.
- virtual void **set_partition** (const double &offset, const double &binwidth, long bins, **Axis** axis)=0
Create a fixed Patton for a valid axis.
- virtual double **get_height** (long indexX, long indexY, long indexZ) const=0
Retrieve the height (value) of a bin.
- virtual void **set_height** (const double &height, long indexX, long indexY, long indexZ)=0
Update the height (value) of a bin.
- virtual **ErrorType** **error_type** (void) const=0
Return the type of errors associated to the bin heights.
- virtual double **get_error** (long indexX, long indexY, long indexZ) const=0
Retrieve the error associated to a bin.
- virtual void **set_error** (const double &error, long indexX, long indexY, long indexZ)=0
Update the error associated to a bin.
- virtual void **get_pmerror** (double &pluserror, double &minuserror, long indexX, long indexY, long indexZ) const=0
Retrieve the plus and minus errors associated to a bin.
- virtual void **set_pmerror** (const double &pluserror, const double &minuserror, long indexX, long indexY, long indexZ)=0
Update the plus and minus errors associated to a bin.
- virtual double **get_mean** (**Axis** axis) const=0
Retrieve the (projected) mean value for a valid axis.
- virtual void **set_mean** (const double &mean, **Axis** axis)=0
Update the (projected) mean value for a valid axis.

- virtual double **get_rms** (**Axis** axis) const=0
Retrieve the (projected) rms value for a valid axis.
- virtual void **set_rms** (const double &rms, **Axis** axis)=0
Update the (projected) rms value for a valid axis.
- virtual void **reset_bins** (void)=0
Reset the bin content.

Static Public Attributes

- const long **Underflow** = 0
Underflow index.
- const long **Overflow** = -1
Overflow index.

5.7.1 Detailed Description

Histogram data interface.

This pure virtual class is used as a base class for the internal histogram data classes.

Definition at line 28 of file OHHistogramData.h.

5.7.2 Member Function Documentation

5.7.2.1 virtual **AxisType** OHHistogramData::axis_type (**Axis** axis) const [pure virtual]

Retrieve type of a valid axis.

This function can be used to determine how many axes that is associated to a histogram. For a one dimensional histogram axis_type(Y) and axis_type(Z) will return NoAxis. For a two dimensional histogram axis_type(Z) will return NoAxis.

Implemented in **OHHistogramData1D** (p. 40).

5.7.2.2 virtual long OHHistogramData::bin_count (**Axis** axis) const [pure virtual]

Retrieve the number of bins for a valid axis.

Returns:

The number of bins for the specified axis, underflow and overflow bins not included.

Implemented in **OHHistogramData1D** (p. 40).

5.7.2.3 `virtual long OHHistogramData::get_annotations (vector< string > & labels, vector< string > & values) const` [pure virtual]

Retrieve annotations.

Returns:

The number of annotations.

Implemented in `OHHistogramData1D` (p. 41).

5.7.2.4 `virtual double OHHistogramData::get_error (long indexX, long indexY, long indexZ) const` [pure virtual]

Retrieve the error associated to a bin.

Underflow and Overflow is valid index values. indexZ is only valid for 3D histograms, indexY is only valid for 2D and 3D histograms. Bin induces start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 33) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type PLUSMINUS_ERROR and the total error is then returned.

Implemented in `OHHistogramData1D` (p. 41).

5.7.2.5 `virtual double OHHistogramData::get_height (long indexX, long indexY, long indexZ) const` [pure virtual]

Retrieve the height (value) of a bin.

Underflow and Overflow is valid index values. indexZ is only valid for 3D histograms, indexY is only valid for 2D and 3D histograms. Bin indexes start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 33) + 1 or Overflow refer to the overflow bin.

Implemented in `OHHistogramData1D` (p. 42).

5.7.2.6 `virtual void OHHistogramData::get_partition (double & offset, double & binwidth, long & bins, Axis axis) const` [pure virtual]

Retrieve a fixed Patton for a valid axis.

This function may only be called if `axis_type()` (p. 33) returns FixedAxis.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

axis Determines for which axis the partition should be retrieved.

Implemented in `OHHistogramData1D` (p. 42).

5.7.2.7 `virtual void OHHistogramData::get_partition (vector< double > & partition, Axis axis) const` [pure virtual]

Retrieve a fixed or variable Patton for a valid axis.

Parameters:

partition A double vector which should be filled with the bin edges.

axis Determines for which axis the partition should be retrieved.

Implemented in **OHHistogramData1D** (p. 43).

5.7.2.8 virtual void OHHistogramData::get_pmerror (double & *pluserror*, double & *minuserror*, long *indexX*, long *indexY*, long *indexZ*) const [pure virtual]

Retrieve the plus and minus errors associated to a bin.

Underflow and Overflow is valid index values. *indexZ* is only valid for 3D histograms, *indexY* is only valid for 2D and 3D histograms. Bin induces start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 33) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type SINGLE_ERROR and the error divided by two is then returned in *pluserror* and *minuserror*.

Implemented in **OHHistogramData1D** (p. 43).

5.7.2.9 virtual void OHHistogramData::reset_bins (void) [pure virtual]

Reset the bin content.

Reset the bin content, this function must be called before the error type associated to the bins is changed.

See also:

set_error() (p. 35) and **set_pmerror()** (p. 36)

Implemented in **OHHistogramData1D** (p. 44).

5.7.2.10 virtual void OHHistogramData::set_annotations (const vector< string > & *labels*, const vector< string > & *values*) [pure virtual]

Update annotations.

The argument vectors must have the same length.

Implemented in **OHHistogramData1D** (p. 44).

5.7.2.11 virtual void OHHistogramData::set_error (const double & *error*, long *indexX*, long *indexY*, long *indexZ*) [pure virtual]

Update the error associated to a bin.

Underflow and Overflow is valid index values. *indexZ* is only valid for 3D histograms, *indexY* is only valid for 2D and 3D histograms. Bin induces start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 33) + 1 or Overflow refer to the overflow bin.

This function may not be used with **set_pmerror()** (p. 36), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by **error_type()** (p. 32) to SINGLE_ERROR.

See also:

`reset_bins()` (p. 35)

Implemented in `OHHistogramData1D` (p. 45).

5.7.2.12 `virtual void OHHistogramData::set_height (const double & height, long indexX, long indexY, long indexZ)` [pure virtual]

Update the height (value) of a bin.

Underflow and Overflow is valid index values. `indexZ` is only valid for 3D histograms, `indexY` is only valid for 2D and 3D histograms. Bin induces start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 33) + 1 or Overflow refer to the overflow bin.

Implemented in `OHHistogramData1D` (p. 45).

5.7.2.13 `virtual void OHHistogramData::set_partition (const double & offset, const double & binwidth, long bins, Axis axis)` [pure virtual]

Create a fixed Patton for a valid axis.

This function invalidates bin contents.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

axis Determines for which axis a new partition should be created.

Implemented in `OHHistogramData1D` (p. 45).

5.7.2.14 `virtual void OHHistogramData::set_partition (const vector< double > & partition, Axis axis)` [pure virtual]

Create a variable Patton for a valid axis.

This function invalidates bin contents.

Parameters:

partition A double vector of bin edges.

axis Determines for which axis a new partition should be created.

Implemented in `OHHistogramData1D` (p. 46).

5.7.2.15 `virtual void OHHistogramData::set_perror (const double & pluserror, const double & minuserror, long indexX, long indexY, long indexZ)` [pure virtual]

Update the plus and minus errors associated to a bin.

Underflow and Overflow is valid index values. `indexZ` is only valid for 3D histograms, `indexY` is only valid for 2D and 3D histograms. Bin induces start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 33) + 1 or Overflow refer to the overflow bin.

This function may not be used with `set_error()` (p. 35), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by `error_type()` (p. 32) to `PLUSMINUS_ERROR`.

See also:

`reset_bins()` (p. 35)

Implemented in `OHHistogramData1D` (p. 47).

5.7.3 Member Data Documentation

5.7.3.1 `const long OHHistogramData::Overflow = -1` [static]

Overflow index.

Index value which can be used to access overflow bins.

See also:

`get_height()` (p. 34), `set_height()` (p. 36), `get_error()` (p. 34), `set_error()` (p. 35), `get_pmerror()` (p. 35) and `set_pmerror()` (p. 36).

Definition at line 66 of file `OHHistogramData.h`.

5.7.3.2 `const long OHHistogramData::Underflow = 0` [static]

Underflow index.

Index value which can be used to access underflow bins.

See also:

`get_height()` (p. 34), `set_height()` (p. 36), `get_error()` (p. 34), `set_error()` (p. 35), `get_pmerror()` (p. 35) and `set_pmerror()` (p. 36).

Definition at line 58 of file `OHHistogramData.h`.

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

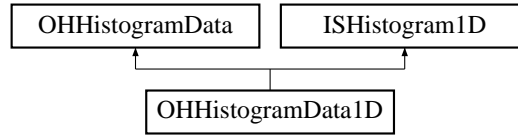
- `OHHistogramData.h`

5.8 OHHistogramData1D Class Reference

One dimensional histogram data.

```
#include <OHHistogramData1D.h>
```

Inheritance diagram for OHHistogramData1D::



Public Methods

- **OHHistogramData1D** ()
Default constructor.
- virtual string **get_title** (void) const
Retrieve the title.
- virtual void **set_title** (const string &title)
Update the title.
- virtual long **get_annotations** (vector< string > &labels, vector< string > &values) const
Retrieve annotations.
- virtual void **set_annotations** (const vector< string > &labels, const vector< string > &values)
Update annotations.
- virtual string **get_label** (**Axis** axis=X) const
Retrieve label of x-axis.
- virtual void **set_label** (const string &label, **Axis** axis=X)
Update label of x-axis.
- virtual **AxisType** **axis_type** (**Axis** axis=X) const
Retrieve axis type.
- virtual long **bin_count** (**Axis** axis=X) const
Retrieve the number of bins for the x-axis.
- virtual void **get_partition** (vector< double > &partition, **Axis** axis=X) const
Retrieve a fixed or variable x-axis partition.
- virtual void **get_partition** (double &offset, double &binwidth, long &bins, **Axis** axis=X) const
Retrieve a fixed x-axis partition.

- virtual void **set_partition** (const vector< double > &partition, **Axis** axis=X)
Create a variable x-axis partition.
- virtual void **set_partition** (const double &offset, const double &binwidth, long bins, **Axis** axis=X)
Create a fixed x-axis partition.
- virtual double **get_height** (long indexX, long indexY=0, long indexZ=0) const
Retrieve the height (value) of a bin.
- virtual void **set_height** (const double &height, long indexX, long indexY=0, long indexZ=0)
Update the height (value) of a bin.
- virtual **ErrorType** **error_type** (void) const
Return the type of errors associated to the bin heights.
- virtual double **get_error** (long indexX, long indexY=0, long indexZ=0) const
Retrieve the error associated to a bin.
- virtual void **set_error** (const double &error, long indexX, long indexY=0, long indexZ=0)
Update the error associated to a bin.
- virtual void **get_pmerror** (double &pluserror, double &minuserror, long indexX, long indexY=0, long indexZ=0) const
Retrieve the plus and minus errors associated to a bin.
- virtual void **set_pmerror** (const double &pluserror, const double &minuserror, long indexX, long indexY=0, long indexZ=0)
Update the plus and minus errors associated to a bin.
- virtual double **get_mean** (**Axis** axis=X) const
Retrieve the mean value.
- virtual void **set_mean** (const double &mean, **Axis** axis=X)
Update the mean value.
- virtual double **get_rms** (**Axis** axis=X) const
Retrieve the rms value.
- virtual void **set_rms** (const double &rms, **Axis** axis=X)
Update the rms value.
- virtual void **reset_bins** (void)
Reset the bin content.

5.8.1 Detailed Description

One dimensional histogram data.

This class provides an interface to one dimensional histogram data and acts as an abstraction layer to the underlying representation of a histogram.

See also:

OHHistogramData (p. 31)

Definition at line 21 of file OHHistogramData1D.h.

5.8.2 Member Function Documentation

5.8.2.1 OHHistogramData1D::AxisType OHHistogramData1D::axis_type (Axis *axis* = X) const [virtual]

Retrieve axis type.

Returns:

NoAxis for Y and Z, else FixedAxis or VARIABLE axis

Implements **OHHistogramData** (p. 33).

Definition at line 47 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_xtype.

```

47                                     {
48     if ( axis == Y || axis == Z ) return NoAxis;
49     if ( is_xtype == IS_FIXED ) return FixedAxis;
50     return VariableAxis;
51 }
```

5.8.2.2 long OHHistogramData1D::bin_count (Axis *axis* = X) const [inline, virtual]

Retrieve the number of bins for the x-axis.

Parameters:

axis Must be X

Returns:

The number of bins for the x-axis, underflow and overflow bins not included.

Implements **OHHistogramData** (p. 33).

Definition at line 36 of file OHHistogramData1D.inl.

References ISHistogram1D::is_xaxis, ISHistogram1D::is_xaxis_size, and ISHistogram1D::is_xtype.

```

36                                     {
37     assert( axis == X );
38     if ( is_xaxis_size > 0 ) {
39         if ( is_xtype == IS_VARIABLE ) {
```

```

40         // We have a vector of edges so we must subtract 1
41         return is_xaxis_size - 1;
42     } else {
43         // Axis vector contains { offset, binwidth, bincount }
44         return static_cast<long>( is_xaxis[ 2 ] );
45     }
46 }
47 return 0;
48 }

```

5.8.2.3 long OHHistogramData1D::get_annotations (vector< string > & labels, vector< string > & values) const [virtual]

Retrieve annotations.

Returns:

The number of annotations.

Implements **OHHistogramData** (p. 34).

Definition at line 13 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_annotations, and ISHistogram1D::is_annotations_size.

```

14                                     {
15     labels.clear( );
16     values.clear( );
17     assert( is_annotations_size % 2 == 0 );
18     for( size_t i = 0; i < ( is_annotations_size / 2 ); i++ ) {
19         labels.push_back( is_annotations[ i*2 ] );
20         values.push_back( is_annotations[ i*2 + 1 ] );
21     }
22     return labels.size( );
23 }

```

5.8.2.4 double OHHistogramData1D::get_error (long indexX, long indexY = 0, long indexZ = 0) const [virtual]

Retrieve the error associated to a bin.

Bin index start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 40) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type PLUSMINUS_ERROR and the total error is then returned.

Implements **OHHistogramData** (p. 34).

Definition at line 161 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_errors, ISHistogram1D::is_errors_size, ISHistogram1D::is_minusererrors, ISHistogram1D::is_minusererrors_size, ISHistogram1D::is_plusererrors, and ISHistogram1D::is_plusererrors_size.

```

163                                     {
164     assert( is_errors_size != 0 ||
165           ( is_plusererrors_size != 0 && is_minusererrors_size != 0 ) );
166     if ( is_errors_size != 0 ) {

```

```

167     return is_errors[ index( indexX ) ];
168 } else {
169     return is_pluserrors[ index( indexX ) ] +
170           is_minusererrors[ index( indexX ) ];
171 }
172 }

```

5.8.2.5 double `OHHistogramData1D::get_height` (long *indexX*, long *indexY* = 0, long *indexZ* = 0) const [inline, virtual]

Retrieve the height (value) of a bin.

Bin index start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 40) + 1 or Overflow refer to the overflow bin.

Implements `OHHistogramData` (p. 34).

Definition at line 54 of file `OHHistogramData1D.inl`.

References `ISHistogram1D::is_bins`, and `ISHistogram1D::is_bins_size`.

```

56                                     {
57     assert( is_bins_size != 0 );
58     return is_bins[ index( indexX ) ];
59 }

```

5.8.2.6 void `OHHistogramData1D::get_partition` (double & *offset*, double & *binwidth*, long & *bins*, Axis *axis* = X) const [virtual]

Retrieve a fixed x-axis partition.

This function may only be called if `axis_type()` (p. 40) returns `FixedAxis`.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

axis Must be X.

Implements `OHHistogramData` (p. 34).

Definition at line 82 of file `OHHistogramData1D.cxx`.

References `ISHistogram1D::is_xaxis`, `ISHistogram1D::is_xaxis_size`, and `ISHistogram1D::is_xtype`.

```

85                                     {
86     assert( axis == X );
87     assert( is_xtype == IS_FIXED );
88     if ( is_xaxis_size > 0 ) {
89         // Axis vector contains { offset, binwidth, bincount }
90         offset = is_xaxis[ 0 ];
91         binwidth = is_xaxis[ 1 ];
92         bins = static_cast<long>( is_xaxis[ 2 ] );
93     } else {
94         offset = binwidth = bins = 0;
95     }
96 }

```

5.8.2.7 void OHHistogramData1D::get_partition (vector< double > & *partition*, Axis *axis* = X) const [virtual]

Retrieve a fixed or variable x-axis partition.

Parameters:

partition A double vector which will be filled with the bin edges.

axis Must be X.

Implements **OHHistogramData** (p. 34).

Definition at line 57 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_xaxis, ISHistogram1D::is_xaxis_size, and ISHistogram1D::is_xtype.

```

58                                     {
59     assert( axis == X );
60     partition.clear( );
61     if ( is_xtype == IS_VARIABLE ) {
62         for ( size_t i = 0; i < is_xaxis_size; i++ ) {
63             partition.push_back( is_xaxis[ i ] );
64         }
65     } else if ( is_xaxis_size > 0 ) {
66         // Axis vector contains { offset, binwidth, bincount }
67         long numbins = static_cast<long>( is_xaxis[ 2 ] );
68         for ( long i = 0; i <= numbins; i++ ) {
69             partition.push_back( is_xaxis[ 0 ] + is_xaxis[ 1 ] * i );
70         }
71     }
72 }
```

5.8.2.8 void OHHistogramData1D::get_pmerror (double & *plusererror*, double & *minusererror*, long *indexX*, long *indexY* = 0, long *indexZ* = 0) const [virtual]

Retrieve the plus and minus errors associated to a bin.

Bin index start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 40) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type SINGLE_ERROR and the error divided by two is then returned in *plusererror* and *minusererror*.

Implements **OHHistogramData** (p. 35).

Definition at line 206 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_errors, ISHistogram1D::is_errors_size, ISHistogram1D::is_minusererrors, ISHistogram1D::is_minusererrors_size, ISHistogram1D::is_plusererrors, and ISHistogram1D::is_plusererrors_size.

```

210                                     {
211     assert( is_errors_size != 0 ||
212            ( is_plusererrors_size != 0 && is_minusererrors_size != 0 ) );
213     if ( is_plusererrors_size != 0 ) {
214         plusererror = is_plusererrors[ index( indexX ) ];
215         minusererror = is_minusererrors[ index( indexX ) ];
216     } else {
217         double half = is_errors[ index( indexX ) ] * 0.5;
```

```

218     pluserror = half;
219     minuserror = half;
220 }
221 }

```

5.8.2.9 void OHHistogramData1D::reset_bins (void) [virtual]

Reset the bin content.

Reset the bin content, this function must be called before the error type associated to the bins is changed.

See also:

`set_error()` (p. 45) and `set_pmerror()` (p. 47)

Implements **OHHistogramData** (p. 35).

Definition at line 228 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_bins, ISHistogram1D::is_bins_size, ISHistogram1D::is_errors, ISHistogram1D::is_errors_size, ISHistogram1D::is_minusererrors, ISHistogram1D::is_minusererrors_size, ISHistogram1D::is_plusererrors, and ISHistogram1D::is_plusererrors_size.

Referenced by `set_partition`.

```

228                                     {
229     if ( is_bins_size != 0 ) delete[] is_bins;
230     if ( is_errors_size != 0 ) delete[] is_errors;
231     if ( is_plusererrors_size != 0 ) delete[] is_plusererrors;
232     if ( is_minusererrors_size != 0 ) delete[] is_minusererrors;
233     is_bins_size = is_errors_size = is_plusererrors_size =
234     is_minusererrors_size = 0;
235 }

```

5.8.2.10 void OHHistogramData1D::set_annotations (const vector< string > & labels, const vector< string > & values) [virtual]

Update annotations.

The argument vectors must have the same length.

Implements **OHHistogramData** (p. 35).

Definition at line 28 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_annotations, and ISHistogram1D::is_annotations_size.

```

29                                     {
30     assert( labels.size( ) == values.size( ) );
31     size_t size = labels.size( ) * 2;
32     if ( is_annotations_size != size ) {
33         if ( is_annotations_size != 0 ) delete[] is_annotations;
34         is_annotations_size = size;
35         is_annotations = new string[ is_annotations_size ];
36     }
37     for ( size_t i = 0; i < labels.size( ); i++ ) {
38         is_annotations[ i*2 ] = labels[ i ];
39         is_annotations[ i*2 + 1 ] = values[ i ];
40     }
41 }

```

5.8.2.11 void OHHistogramData1D::set_error (const double & *error*, long *indexX*, long *indexY* = 0, long *indexZ* = 0) [inline, virtual]

Update the error associated to a bin.

Bin index start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 40) + 1 or Overflow refer to the overflow bin.

This function may not be used with `set_perror()` (p. 47), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by `error_type()` (p. 39) to `SINGLE_ERROR`.

See also:

`reset_bins()` (p. 44)

Implements **OHHistogramData** (p. 35).

Definition at line 85 of file OHHistogramData1D.inl.

References `ISHistogram1D::is_errors`, `ISHistogram1D::is_minusererrors_size`, and `ISHistogram1D::is_plusererrors_size`.

```

88                                     {
89     assert( is_plusererrors_size == 0 && is_minusererrors_size == 0 );
90     assert( error >= 0 );
91     alloc_errors( );
92     is_errors[ index( indexX ) ] = error;
93 }
```

5.8.2.12 void OHHistogramData1D::set_height (const double & *height*, long *indexX*, long *indexY* = 0, long *indexZ* = 0) [inline, virtual]

Update the height (value) of a bin.

Bin index start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 40) + 1 or Overflow refer to the overflow bin.

Implements **OHHistogramData** (p. 36).

Definition at line 65 of file OHHistogramData1D.inl.

References `ISHistogram1D::is_bins`.

```

68                                     {
69     alloc_heights( );
70     is_bins[ index( indexX ) ] = height;
71 }
```

5.8.2.13 void OHHistogramData1D::set_partition (const double & *offset*, const double & *binwidth*, long *bins*, Axis *axis* = X) [virtual]

Create a fixed x-axis partition.

This function invalidates bin contents.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

axis Must be X.

Implements **OHHistogramData** (p. 36).

Definition at line 126 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_xaxis, ISHistogram1D::is_xaxis_size, ISHistogram1D::is_xtype, and reset_bins.

```

129                                     {
130     assert( axis == X );
131     assert( bins > 0 );
132     assert( binwidth > 0 );
133     reset_bins( );
134     if ( is_xaxis_size != 3 ) {
135         if ( is_xaxis_size != 0 ) delete[] is_xaxis;
136         is_xaxis_size = 3;
137         is_xaxis = new double[ is_xaxis_size ];
138     }
139     is_xaxis[ 0 ] = offset;
140     is_xaxis[ 1 ] = binwidth;
141     is_xaxis[ 2 ] = bins;
142     is_xtype = IS_FIXED;
143 }
```

5.8.2.14 void OHHistogramData1D::set_partition (const vector< double > & *partition*, Axis *axis* = X) [virtual]

Create a variable x-axis partition.

This function invalidates bin contents.

Parameters:

partition A double vector of bin edges.

axis Must be X.

Implements **OHHistogramData** (p. 36).

Definition at line 103 of file OHHistogramData1D.cxx.

References ISHistogram1D::is_xaxis, ISHistogram1D::is_xaxis_size, ISHistogram1D::is_xtype, and reset_bins.

```

104                                     {
105     assert( axis == X );
106     assert( partition.size( ) > 1 );
107     reset_bins( );
108     if ( is_xaxis_size != partition.size( ) ) {
109         if ( is_xaxis_size != 0 ) delete[] is_xaxis;
110         is_xaxis_size = partition.size( );
111         is_xaxis = new double[ is_xaxis_size ];
112     }
113     for ( size_t i = 0; i < is_xaxis_size; i++ ) {
114         is_xaxis[ i ] = partition[ i ];
115     }
116     is_xtype = IS_VARIABLE;
117 }
```

5.8.2.15 void OHHistogramData1D::set_pmerror (const double & *plusererror*, const double & *minusererror*, long *indexX*, long *indexY* = 0, long *indexZ* = 0) [inline, virtual]

Update the plus and minus errors associated to a bin.

Bin index start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 40) + 1 or Overflow refer to the overflow bin.

This function may not be used with `set_error()` (p. 45), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by `error_type()` (p. 39) to PLUSMINUS_ERROR.

See also:

`reset_bins()` (p. 44)

Implements `OHHistogramData` (p. 36).

Definition at line 186 of file OHHistogramData1D.cxx.

References `ISHistogram1D::is_errors_size`, `ISHistogram1D::is_minusererrors`, and `ISHistogram1D::is_plusererrors`.

```

190                                     {
191     assert( plusererror >= 0 && minusererror >= 0 );
192     assert( is_errors_size == 0 );
193     alloc_pmerrors( );
194     is_plusererrors[ index( indexX ) ] = plusererror;
195     is_minusererrors[ index( indexX ) ] = minusererror;
196 }
```

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

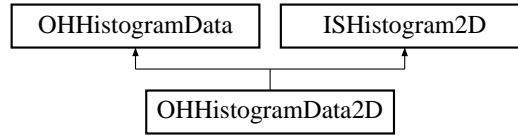
- `OHHistogramData1D.h`
- `OHHistogramData1D.inl`
- `OHHistogramData1D.cxx`

5.9 OHHistogramData2D Class Reference

Two dimensional histogram data.

```
#include <OHHistogramData2D.h>
```

Inheritance diagram for OHHistogramData2D::



Public Methods

- **OHHistogramData2D** ()
Default constructor.
- virtual string **get_title** (void) const
Retrieve the title.
- virtual void **set_title** (const string &title)
Update the title.
- virtual long **get_annotations** (vector< string > &labels, vector< string > &values) const
Retrieve annotations.
- virtual void **set_annotations** (const vector< string > &labels, const vector< string > &values)
Update annotations.
- virtual string **get_label** (**Axis** axis) const
Retrieve label of x or y-axis.
- virtual void **set_label** (const string &label, **Axis** axis)
Update label of x or y-axis.
- virtual **AxisType** **axis_type** (**Axis** axis) const
Retrieve type of the x or y-axis.
- virtual long **bin_count** (**Axis** axis) const
Retrieve the number of bins for the x or y-axis.
- virtual void **get_partition** (vector< double > &partition, **Axis** axis) const
Retrieve a fixed or variable partition for the x or y-axis.
- virtual void **get_partition** (double &offset, double &binwidth, long &bins, **Axis** axis) const
Retrieve a fixed partition for the x or y-axis.

- virtual void **set_partition** (const vector< double > &partition, **Axis** axis)
Create a variable partition for the x or y-axis.
- virtual void **set_partition** (const double &offset, const double &binwidth, long bins, **Axis** axis)
Create a fixed partition for the x or y-axis.
- virtual double **get_height** (long indexX, long indexY, long indexZ=0) const
Retrieve the height (value) of a bin.
- virtual void **set_height** (const double &height, long indexX, long indexY, long indexZ=0)
Update the height (value) of a bin.
- virtual **ErrorType** **error_type** (void) const
Return the type of errors associated to the bin heights.
- virtual double **get_error** (long indexX, long indexY, long indexZ=0) const
Retrieve the error associated to a bin.
- virtual void **set_error** (const double &error, long indexX, long indexY, long indexZ=0)
Update the error associated to a bin.
- virtual void **get_pmerror** (double &pluserror, double &minuserror, long indexX, long indexY, long indexZ=0) const
Retrieve the plus and minus errors associated to a bin.
- virtual void **set_pmerror** (const double &pluserror, const double &minuserror, long indexX, long indexY, long indexZ=0)
Update the plus and minus errors associated to a bin.
- virtual double **get_mean** (**Axis** axis) const
Retrieve the (projected) mean value for the x or y-axis.
- virtual void **set_mean** (const double &mean, **Axis** axis)
Update the (projected) mean value for the x or y-axis.
- virtual double **get_rms** (**Axis** axis) const
Retrieve the (projected) rms value for the x or y-axis.
- virtual void **set_rms** (const double &rms, **Axis** axis)
Update the (projected) rms value for the x or y-axis.
- virtual void **reset_bins** (void)
Reset the bin content.

5.9.1 Detailed Description

Two dimensional histogram data.

This class provides an interface to two dimensional histogram data and acts as an abstraction layer to the underlying representation of a histogram.

See also:

OHHistogramData (p. 31)

Definition at line 19 of file OHHistogramData2D.h.

5.9.2 Member Function Documentation

5.9.2.1 OHHistogramData2D::AxisType OHHistogramData2D::axis_type (Axis *axis*) const [virtual]

Retrieve type of the x or y-axis.

Returns:

NoAxis for Z, else FixedAxis or VARIABLE axis

Implements **OHHistogramData** (p. 33).

Definition at line 60 of file OHHistogramData2D.cxx.

References OHHistogramData::AxisType, ISHistogram2D::is_xtype, and ISHistogram2D::is_ytype.

```

60                                     {
61     if ( axis == Z ) return NoAxis;
62     if ( axis == X && is_xtype == IS_FIXED ) return FixedAxis;
63     if ( axis == Y && is_ytype == IS_FIXED ) return FixedAxis;
64     return VariableAxis;
65 }
```

5.9.2.2 long OHHistogramData2D::bin_count (Axis *axis*) const [virtual]

Retrieve the number of bins for the x or y-axis.

Parameters:

axis Must be X or Y

Returns:

The number of bins for the specified axis, underflow and overflow bins not included.

Implements **OHHistogramData** (p. 33).

Definition at line 72 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_xaxis, ISHistogram2D::is_xaxis_size, ISHistogram2D::is_xtype, ISHistogram2D::is_yaxis, ISHistogram2D::is_yaxis_size, and ISHistogram2D::is_ytype.

```

72                                     {
73     assert( axis == X || axis == Y );
74     double **ax;
```

```

75     size_t *sz;
76     is_xtype_E *ty;
77     if ( axis == X ) {
78         ax = const_cast<double**>( &is_xaxis );
79         sz = const_cast<size_t*>( &is_xaxis_size );
80         ty = const_cast<is_xtype_E*>( &is_xtype );
81     } else {
82         ax = const_cast<double**>( &is_yaxis );
83         sz = const_cast<size_t*>( &is_yaxis_size );
84         ty = const_cast<is_xtype_E*>( &is_ytype );
85     }
86     if ( *sz > 0 ) {
87         if ( *ty == IS_VARIABLE ) {
88             // We have a vector of edges so we must subtract 1
89             return *sz - 1;
90         } else {
91             // Axis vector contains { offset, binwidth, bincount }
92             return static_cast<long>( (*ax)[ 2 ] );
93         }
94     }
95     return 0;
96 }

```

5.9.2.3 long OHHistogramData2D::get_annotations (vector< string > & labels, vector< string > & values) const [virtual]

Retrieve annotations.

Returns:

The number of annotations.

Implements **OHHistogramData** (p. 34).

Definition at line 12 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_annotations, and ISHistogram2D::is_annotations_size.

```

13                                     {
14     labels.clear( );
15     values.clear( );
16     assert( is_annotations_size % 2 == 0 );
17     for( size_t i = 0; i < ( is_annotations_size / 2 ); i++ ) {
18         labels.push_back( is_annotations[ i*2      ] );
19         values.push_back( is_annotations[ i*2 + 1 ] );
20     }
21     return labels.size( );
22 }

```

5.9.2.4 double OHHistogramData2D::get_error (long indexX, long indexY, long indexZ = 0) const [virtual]

Retrieve the error associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 50) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type PLUSMINUS_ERROR and the total error is then returned.

Implements **OHHistogramData** (p. 34).

Definition at line 272 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_errors, ISHistogram2D::is_errors_size, ISHistogram2D::is_minusererrors, ISHistogram2D::is_minusererrors_size, ISHistogram2D::is_plusererrors, and ISHistogram2D::is_plusererrors_size.

```

274                                     {
275     assert( is_errors_size != 0 ||
276           ( is_plusererrors_size != 0 && is_minusererrors_size != 0 ) );
277     if ( is_errors_size != 0 ) {
278         return is_errors[ index( indexX, indexY ) ];
279     } else {
280         return is_plusererrors[ index( indexX, indexY ) ] +
281               is_minusererrors[ index( indexX, indexY ) ];
282     }
283 }
```

5.9.2.5 double OHHistogramData2D::get_height (long *indexX*, long *indexY*, long *indexZ* = 0) const [virtual]

Retrieve the height (value) of a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 50) + 1 or Overflow refer to the overflow bin.

Implements **OHHistogramData** (p. 34).

Definition at line 237 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_bins, and ISHistogram2D::is_bins_size.

```

239                                     {
240     assert( is_bins_size != 0 );
241     return is_bins[ index( indexX, indexY ) ];
242 }
```

5.9.2.6 void OHHistogramData2D::get_partition (double & *offset*, double & *binwidth*, long & *bins*, Axis *axis*) const [virtual]

Retrieve a fixed partition for the x or y-axis.

This function may only be called if **axis_type()** (p. 50) returns FixedAxis.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

axis Must be X or Y.

Implements **OHHistogramData** (p. 34).

Definition at line 139 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_xaxis, ISHistogram2D::is_xaxis_size, ISHistogram2D::is_xtype, ISHistogram2D::is_yaxis, ISHistogram2D::is_yaxis_size, and ISHistogram2D::is_ytype.

```

142                                     {
143     assert( axis == X || axis == Y );
144     double **ax;
145     size_t *sz;
146     if ( axis == X ) {
147         assert( is_xtype == IS_FIXED );
148         ax = const_cast<double**>( &is_xaxis );
149         sz = const_cast<size_t*>( &is_xaxis_size );
150     } else {
151         assert( is_ytype == IS_FIXED );
152         ax = const_cast<double**>( &is_yaxis );
153         sz = const_cast<size_t*>( &is_yaxis_size );
154     }
155     if ( *sz > 0 ) {
156         assert( *sz == 3 );
157         // Axis vector contains { offset, binwidth, bincount }
158         offset = (*ax)[ 0 ];
159         binwidth = (*ax)[ 1 ];
160         bins = static_cast<long>( (*ax)[ 2 ] );
161     } else {
162         offset = binwidth = bins = 0;
163     }
164 }

```

5.9.2.7 void OHHistogramData2D::get_partition (vector< double > & *partition*, Axis *axis*) const [virtual]

Retrieve a fixed or variable partition for the x or y-axis.

Parameters:

- partition* A double vector which will be filled with the bin edges.
- axis* Must be X or Y.

Implements **OHHistogramData** (p. 34).

Definition at line 102 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_xaxis, ISHistogram2D::is_xaxis_size, ISHistogram2D::is_xtype, ISHistogram2D::is_yaxis, ISHistogram2D::is_yaxis_size, and ISHistogram2D::is_ytype.

```

103                                     {
104     assert( axis == X || axis == Y );
105     partition.clear( );
106     double **ax;
107     size_t *sz;
108     is_xtype_E *ty;
109     if ( axis == X ) {
110         ax = const_cast<double**>( &is_xaxis );
111         sz = const_cast<size_t*>( &is_xaxis_size );
112         ty = const_cast<is_xtype_E*>( &is_xtype );
113     } else {
114         ax = const_cast<double**>( &is_yaxis );
115         sz = const_cast<size_t*>( &is_yaxis_size );
116         ty = const_cast<is_xtype_E*>( &is_ytype );
117     }
118     if ( *ty == IS_VARIABLE ) {
119         for ( size_t i = 0; i < *sz; i++ ) {
120             partition.push_back( (*ax)[ i ] );
121         }
122     } else if ( *sz > 0 ) {

```

```

123     // Axis vector contains { offset, binwidth, bincount }
124     long numbins = static_cast<long>( (*ax)[ 2 ] );
125     for ( long i = 0; i <= numbins; i++ ) {
126         partition.push_back( (*ax)[ 0 ] + (*ax)[ 1 ] * i );
127     }
128 }
129 }

```

5.9.2.8 void OHHistogramData2D::get_pmerror (double & *plusererror*, double & *minusererror*, long *indexX*, long *indexY*, long *indexZ* = 0) const [virtual]

Retrieve the plus and minus errors associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 50) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type SINGLE_ERROR and the error divided by two is then returned in *plusererror* and *minusererror*.

Implements **OHHistogramData** (p. 35).

Definition at line 315 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_errors, ISHistogram2D::is_errors_size, ISHistogram2D::is_minusererrors, ISHistogram2D::is_minusererrors_size, ISHistogram2D::is_plusererrors, and ISHistogram2D::is_plusererrors_size.

```

319                                     {
320     assert( is_errors_size != 0 ||
321            ( is_plusererrors_size != 0 && is_minusererrors_size != 0 ) );
322     if ( is_plusererrors_size != 0 ) {
323         plusererror = is_plusererrors[ index( indexX, indexY ) ];
324         minusererror = is_minusererrors[ index( indexX, indexY ) ];
325     } else {
326         double half = is_errors[ index( indexX, indexY ) ] * 0.5;
327         plusererror = half;
328         minusererror = half;
329     }
330 }

```

5.9.2.9 void OHHistogramData2D::reset_bins (void) [virtual]

Reset the bin content.

Reset the bin content, this function must be called before the error type associated to the bins is changed.

See also:

set_error() (p. 55) and **set_pmerror()** (p. 58)

Implements **OHHistogramData** (p. 35).

Definition at line 391 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_bins, ISHistogram2D::is_bins_size, ISHistogram2D::is_errors, ISHistogram2D::is_errors_size, ISHistogram2D::is_minusererrors, ISHistogram2D::is_minusererrors_size, ISHistogram2D::is_plusererrors, and ISHistogram2D::is_plusererrors_size.

Referenced by **set_partition**.

```

391                                     {
392     if ( is_bins_size != 0 ) delete[] is_bins;
393     if ( is_errors_size != 0 ) delete[] is_errors;
394     if ( is_plusererrors_size != 0 ) delete[] is_plusererrors;
395     if ( is_minusererrors_size != 0 ) delete[] is_minusererrors;
396     is_bins_size = is_errors_size = is_plusererrors_size =
397     is_minusererrors_size = 0;
398 }

```

5.9.2.10 void OHHistogramData2D::set_annotations (const vector< string > & labels, const vector< string > & values) [virtual]

Update annotations.

The argument vectors must have the same length.

Implements **OHHistogramData** (p. 35).

Definition at line 27 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_annotations, and ISHistogram2D::is_annotations_size.

```

28                                     {
29     assert( labels.size( ) == values.size( ) );
30     size_t size = labels.size( ) * 2;
31     if ( is_annotations_size != size ) {
32         if ( is_annotations_size != 0 ) delete[] is_annotations;
33         is_annotations_size = size;
34         is_annotations = new string[ is_annotations_size ];
35     }
36     for ( size_t i = 0; i < labels.size( ); i++ ) {
37         is_annotations[ i*2 ] = labels[ i ];
38         is_annotations[ i*2 + 1 ] = values[ i ];
39     }
40 }

```

5.9.2.11 void OHHistogramData2D::set_error (const double & error, long indexX, long indexY, long indexZ = 0) [virtual]

Update the error associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 50) + 1 or Overflow refer to the overflow bin.

This function may not be used with **set_perror()** (p. 58), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by **error_type()** (p. 49) to SINGLE_ERROR.

See also:

reset_bins() (p. 54)

Implements **OHHistogramData** (p. 35).

Definition at line 297 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_errors, ISHistogram2D::is_minusererrors_size, and ISHistogram2D::is_plusererrors_size.

```

300                                     {
301     assert( is_pluseerrors_size == 0 && is_minuseerrors_size == 0 );
302     assert( error >= 0 );
303     alloc_errors( );
304     is_errors[ index( indexX, indexY ) ] = error;
305 }

```

5.9.2.12 void OHHistogramData2D::set_height (const double & *height*, long *indexX*, long *indexY*, long *indexZ* = 0) [virtual]

Update the height (value) of a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 50) + 1 or Overflow refer to the overflow bin.

Implements **OHHistogramData** (p. 36).

Definition at line 248 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_bins.

```

251                                     {
252     alloc_heights( );
253     is_bins[ index( indexX, indexY ) ] = height;
254 }

```

5.9.2.13 void OHHistogramData2D::set_partition (const double & *offset*, const double & *binwidth*, long *bins*, Axis *axis*) [virtual]

Create a fixed partition for the x or y-axis.

This function invalidates bin contents.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

axis Must be X or Y.

Implements **OHHistogramData** (p. 36).

Definition at line 204 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_xaxis, ISHistogram2D::is_xaxis_size, ISHistogram2D::is_xtype, ISHistogram2D::is_yaxis, ISHistogram2D::is_yaxis_size, ISHistogram2D::is_ytype, and reset_bins.

```

207                                     {
208     assert( axis == X || axis == Y );
209     assert( bins > 0 );
210     assert( binwidth > 0 );
211     reset_bins( );
212     double **ax;
213     size_t *sz;
214     if ( axis == X ) {
215         ax = &is_xaxis;
216         sz = &is_xaxis_size;
217         is_xtype = IS_FIXED;

```

```

218     } else {
219         ax = &is_yaxis;
220         sz = &is_yaxis_size;
221         is_ytype = IS_FIXED;
222     }
223     if ( *sz != 3 ) {
224         if ( *sz != 0 ) delete[] *ax;
225         *sz = 3;
226         *ax = new double[ *sz ];
227     }
228     (*ax)[ 0 ] = offset;
229     (*ax)[ 1 ] = binwidth;
230     (*ax)[ 2 ] = bins;
231 }

```

5.9.2.14 void OHHistogramData2D::set_partition (const vector< double > & *partition*, Axis *axis*) [virtual]

Create a variable partition for the x or y-axis.

This function invalidates bin contents.

Parameters:

partition A double vector of bin edges.

axis Must be X or Y.

Implements **OHHistogramData** (p. 36).

Definition at line 171 of file OHHistogramData2D.cxx.

References ISHistogram2D::is_xaxis, ISHistogram2D::is_xaxis_size, ISHistogram2D::is_xtype, ISHistogram2D::is_yaxis, ISHistogram2D::is_yaxis_size, ISHistogram2D::is_ytype, and reset_bins.

```

172                                     {
173     assert( axis == X || axis == Y );
174     assert( partition.size( ) > 1 );
175     reset_bins( );
176     double **ax;
177     size_t *sz;
178     if ( axis == X ) {
179         ax = &is_xaxis;
180         sz = &is_xaxis_size;
181         is_xtype = IS_VARIABLE;
182     } else {
183         ax = &is_yaxis;
184         sz = &is_yaxis_size;
185         is_ytype = IS_VARIABLE;
186     }
187     if ( *sz != partition.size( ) ) {
188         if ( *sz != 0 ) delete[] *ax;
189         *sz = partition.size( );
190         *ax = new double[ *sz ];
191     }
192     for ( size_t i = 0; i < *sz; i++ ) {
193         (*ax)[ i ] = partition[ i ];
194     }
195 }

```

5.9.2.15 `void OHHistogramData2D::set_pmerror (const double & plusererror,
const double & minusererror, long indexX, long indexY, long indexZ = 0)
[virtual]`

Update the plus and minus errors associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 50) + 1 or Overflow refer to the overflow bin.

This function may not be used with `set_error()` (p. 55), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by `error_type()` (p. 49) to `PLUSMINUS_ERROR`.

See also:

`reset_bins()` (p. 54)

Implements `OHHistogramData` (p. 36).

Definition at line 344 of file `OHHistogramData2D.cxx`.

References `ISHistogram2D::is_errors_size`, `ISHistogram2D::is_minusererrors`, and `ISHistogram2D::is_plusererrors`.

```

348                                     {
349     assert( is_errors_size == 0 );
350     assert( plusererror >= 0 && minusererror >= 0 );
351     alloc_pmerrors( );
352     is_plusererrors[ index( indexX, indexY ) ] = plusererror;
353     is_minusererrors[ index( indexX, indexY ) ] = minusererror;
354 }
```

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

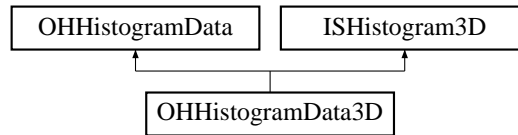
- `OHHistogramData2D.h`
- `OHHistogramData2D.inl`
- `OHHistogramData2D.cxx`

5.10 OHHistogramData3D Class Reference

Three dimensional histogram data.

```
#include <OHHistogramData3D.h>
```

Inheritance diagram for OHHistogramData3D::



Public Methods

- **OHHistogramData3D** ()
Default constructor.
- virtual string **get_title** (void) const
Retrieve the title.
- virtual void **set_title** (const string &title)
Update the title.
- virtual long **get_annotatations** (vector< string > &labels, vector< string > &values) const
Retrieve annotations.
- virtual void **set_annotatations** (const vector< string > &labels, const vector< string > &values)
Update annotations.
- virtual string **get_label** (**Axis** axis) const
Retrieve label of an axis.
- virtual void **set_label** (const string &label, **Axis** axis)
Update label of an axis.
- virtual **AxisType** **axis_type** (**Axis** axis) const
Retrieve type of an axis.
- virtual long **bin_count** (**Axis** axis) const
Retrieve the number of bins for an axis.
- virtual void **get_partition** (vector< double > &partition, **Axis** axis) const
Retrieve a fixed or variable partition for an axis.
- virtual void **get_partition** (double &offset, double &binwidth, long &bins, **Axis** axis) const
Retrieve a fixed partition for an axis.

- virtual void **set_partition** (const vector< double > &partition, **Axis** axis)
Create a variable partition for an axis.
- virtual void **set_partition** (const double &offset, const double &binwidth, long bins, **Axis** axis)
Create a fixed partition for an axis.
- virtual double **get_height** (long indexX, long indexY, long indexZ) const
Retrieve the height (value) of a bin.
- virtual void **set_height** (const double &height, long indexX, long indexY, long indexZ)
Update the height (value) of a bin.
- virtual **ErrorType error_type** (void) const
Return the type of errors associated to the bin heights.
- virtual double **get_error** (long indexX, long indexY, long indexZ) const
Retrieve the error associated to a bin.
- virtual void **set_error** (const double &error, long indexX, long indexY, long indexZ)
Update the error associated to a bin.
- virtual void **get_pmerror** (double &pluserror, double &minuserror, long indexX, long indexY, long indexZ) const
Retrieve the plus and minus errors associated to a bin.
- virtual void **set_pmerror** (const double &pluserror, const double &minuserror, long indexX, long indexY, long indexZ)
Update the plus and minus errors associated to a bin.
- virtual double **get_mean** (**Axis** axis) const
Retrieve the (projected) mean value an axis.
- virtual void **set_mean** (const double &mean, **Axis** axis)
Update the (projected) mean value for an axis.
- virtual double **get_rms** (**Axis** axis) const
Retrieve the (projected) rms value for an axis.
- virtual void **set_rms** (const double &rms, **Axis** axis)
Update the (projected) rms value for an axis.
- virtual void **reset_bins** (void)
Reset the bin content.

5.10.1 Detailed Description

Three dimensional histogram data.

This class provides an interface to three dimensional histogram data and acts as an abstraction layer to the underlying representation of a histogram.

See also:

OHHistogramData (p. 31)

Definition at line 20 of file OHHistogramData3D.h.

5.10.2 Member Function Documentation

5.10.2.1 OHHistogramData3D::AxisType OHHistogramData3D::axis_type (Axis *axis*) const [virtual]

Retrieve type of an axis.

Returns:

FixedAxis or VARIABLE axis

Implements **OHHistogramData** (p. 33).

Definition at line 66 of file OHHistogramData3D.cxx.

References **OHHistogramData::AxisType**, **ISHistogram3D::is_xtype**, **ISHistogram3D::is_ytype**, and **ISHistogram3D::is_ztype**.

```

66                                     {
67     if ( axis == X && is_xtype == IS_FIXED ) return FixedAxis;
68     if ( axis == Y && is_ytype == IS_FIXED ) return FixedAxis;
69     if ( axis == Z && is_ztype == IS_FIXED ) return FixedAxis;
70     return VariableAxis;
71 }
```

5.10.2.2 long OHHistogramData3D::bin_count (Axis *axis*) const [virtual]

Retrieve the number of bins for an axis.

Returns:

The number of bins for the specified axis, underflow and overflow bins not included.

Implements **OHHistogramData** (p. 33).

Definition at line 77 of file OHHistogramData3D.cxx.

References **ISHistogram3D::is_xaxis**, **ISHistogram3D::is_xaxis_size**, **ISHistogram3D::is_xtype**, **ISHistogram3D::is_yaxis**, **ISHistogram3D::is_yaxis_size**, **ISHistogram3D::is_ytype**, **ISHistogram3D::is_zaxis**, **ISHistogram3D::is_zaxis_size**, and **ISHistogram3D::is_ztype**.

```

77                                     {
78     double **ax;
79     size_t *sz;
80     is_xtype_E *ty;
```

```

81     if ( axis == X ) {
82         ax = const_cast<double**>( &is_xaxis );
83         sz = const_cast<size_t*>( &is_xaxis_size );
84         ty = const_cast<is_xtype_E*>( &is_xtype );
85     } else if ( axis == Y ) {
86         ax = const_cast<double**>( &is_yaxis );
87         sz = const_cast<size_t*>( &is_yaxis_size );
88         ty = const_cast<is_xtype_E*>( &is_ytype );
89     } else {
90         ax = const_cast<double**>( &is_zaxis );
91         sz = const_cast<size_t*>( &is_zaxis_size );
92         ty = const_cast<is_xtype_E*>( &is_ztype );
93     }
94     if ( *sz > 0 ) {
95         if ( *ty == IS_VARIABLE ) {
96             // We have a vector of edges so we must subtract 1
97             return *sz - 1;
98         } else {
99             // Axis vector contains { offset, binwidth, bincount }
100            return static_cast<long>( (*ax)[ 2 ] );
101        }
102    }
103    return 0;
104 }

```

5.10.2.3 long OHHistogramData3D::get_annotations (vector< string > & labels, vector< string > & values) const [virtual]

Retrieve annotations.

Returns:

The number of annotations.

Implements **OHHistogramData** (p. 34).

Definition at line 12 of file OHHistogramData3D.cxx.

References ISHistogram3D::is_annotations, and ISHistogram3D::is_annotations_size.

```

13                                     {
14     labels.clear( );
15     values.clear( );
16     assert( is_annotations_size % 2 == 0 );
17     for( size_t i = 0; i < ( is_annotations_size / 2 ); i++ ) {
18         labels.push_back( is_annotations[ i*2 ] );
19         values.push_back( is_annotations[ i*2 + 1 ] );
20     }
21     return labels.size( );
22 }

```

5.10.2.4 double OHHistogramData3D::get_error (long indexX, long indexY, long indexZ) const [virtual]

Retrieve the error associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 61) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type PLUSMINUS_ERROR and the total error is then returned.

Implements **OHHistogramData** (p. 34).

Definition at line 289 of file OHHistogramData3D.cxx.

References ISHistogram3D::is_errors, ISHistogram3D::is_errors_size, ISHistogram3D::is_minusererrors, ISHistogram3D::is_minusererrors_size, ISHistogram3D::is_plusererrors, and ISHistogram3D::is_plusererrors_size.

```

291                                     {
292     assert( is_errors_size != 0 ||
293            ( is_plusererrors_size != 0 && is_minusererrors_size != 0 ) );
294     if ( is_errors_size != 0 ) {
295         return is_errors[ index( indexX, indexY, indexZ ) ];
296     } else {
297         return is_plusererrors[ index( indexX, indexY, indexZ ) ] +
298                is_minusererrors[ index( indexX, indexY, indexZ ) ];
299     }
300 }
```

5.10.2.5 double OHHistogramData3D::get_height (long *indexX*, long *indexY*, long *indexZ*) const [virtual]

Retrieve the height (value) of a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 61) + 1 or Overflow refer to the overflow bin.

Implements **OHHistogramData** (p. 34).

Definition at line 254 of file OHHistogramData3D.cxx.

References ISHistogram3D::is_bins, and ISHistogram3D::is_bins_size.

```

256                                     {
257     assert( is_bins_size != 0 );
258     return is_bins[ index( indexX, indexY, indexZ ) ];
259 }
```

5.10.2.6 void OHHistogramData3D::get_partition (double & *offset*, double & *binwidth*, long & *bins*, Axis *axis*) const [virtual]

Retrieve a fixed partition for an axis.

This function may only be called if **axis_type()** (p. 61) returns FixedAxis.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

Implements **OHHistogramData** (p. 34).

Definition at line 149 of file OHHistogramData3D.cxx.

References ISHistogram3D::is_xaxis, ISHistogram3D::is_xaxis_size, ISHistogram3D::is_xtype, ISHistogram3D::is_yaxis, ISHistogram3D::is_yaxis_size, ISHistogram3D::is_ytype, ISHistogram3D::is_zaxis, ISHistogram3D::is_zaxis_size, and ISHistogram3D::is_ztype.

```

152                                     {
153     double **ax;
154     size_t *sz;
155     if ( axis == X ) {
156         assert( is_xtype == IS_FIXED );
157         ax = const_cast<double**>( &is_xaxis );
158         sz = const_cast<size_t*>( &is_xaxis_size );
159     } else if ( axis == Y ) {
160         assert( is_ytype == IS_FIXED );
161         ax = const_cast<double**>( &is_yaxis );
162         sz = const_cast<size_t*>( &is_yaxis_size );
163     } else {
164         assert( is_ztype == IS_FIXED );
165         ax = const_cast<double**>( &is_zaxis );
166         sz = const_cast<size_t*>( &is_zaxis_size );
167     }
168     if ( *sz > 0 ) {
169         assert( *sz == 3 );
170         // Axis vector contains { offset, binwidth, bincount }
171         offset = (*ax)[ 0 ];
172         binwidth = (*ax)[ 1 ];
173         bins = static_cast<long>( (*ax)[ 2 ] );
174     } else {
175         offset = binwidth = bins = 0;
176     }
177 }

```

5.10.2.7 void OHHistogramData3D::get_partition (vector< double > & *partition*, Axis *axis*) const [virtual]

Retrieve a fixed or variable partition for an axis.

Parameters:

partition A double vector which will be filled with the bin edges.

axis Must be X or Y.

Implements **OHHistogramData** (p. 34).

Definition at line 110 of file OHHistogramData3D.cxx.

References ISHistogram3D::is_xaxis, ISHistogram3D::is_xaxis_size, ISHistogram3D::is_xtype, ISHistogram3D::is_yaxis, ISHistogram3D::is_yaxis_size, ISHistogram3D::is_ytype, ISHistogram3D::is_zaxis, ISHistogram3D::is_zaxis_size, and ISHistogram3D::is_ztype.

```

111                                     {
112     partition.clear( );
113     double **ax;
114     size_t *sz;
115     is_xtype_E *ty;
116     if ( axis == X ) {
117         ax = const_cast<double**>( &is_xaxis );
118         sz = const_cast<size_t*>( &is_xaxis_size );
119         ty = const_cast<is_xtype_E*>( &is_xtype );
120     } else if ( axis == Y ) {
121         ax = const_cast<double**>( &is_yaxis );

```

```

122     sz = const_cast<size_t*>( &is_yaxis_size );
123     ty = const_cast<is_xtype_E*>( &is_ytype );
124 } else {
125     ax = const_cast<double**>( &is_zaxis );
126     sz = const_cast<size_t*>( &is_zaxis_size );
127     ty = const_cast<is_xtype_E*>( &is_ztype );
128 }
129 if ( *ty == IS_VARIABLE ) {
130     for ( size_t i = 0; i < *sz; i++ ) {
131         partition.push_back( (*ax)[ i ] );
132     }
133 } else if ( *sz > 0 ) {
134     // Axis vector contains { offset, binwidth, bincount }
135     long numbins = static_cast<long>( (*ax)[ 2 ] );
136     for ( long i = 0; i <= numbins; i++ ) {
137         partition.push_back( (*ax)[ 0 ] + (*ax)[ 1 ] * i );
138     }
139 }
140 }

```

5.10.2.8 void OHHistogramData3D::get_pmerror (double & *plusererror*, double & *minusererror*, long *indexX*, long *indexY*, long *indexZ*) const [virtual]

Retrieve the plus and minus errors associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 61) + 1 or Overflow refer to the overflow bin.

This function is valid also for histograms with errors of type SINGLE_ERROR and the error divided by two is then returned in `plusererror` and `minusererror`.

Implements **OHHistogramData** (p. 35).

Definition at line 332 of file OHHistogramData3D.cxx.

References `ISHistogram3D::is_errors`, `ISHistogram3D::is_errors_size`, `ISHistogram3D::is_minusererrors`, `ISHistogram3D::is_minusererrors_size`, `ISHistogram3D::is_plusererrors`, and `ISHistogram3D::is_plusererrors_size`.

```

336                                     {
337     assert( is_errors_size != 0 ||
338            ( is_plusererrors_size != 0 && is_minusererrors_size != 0 ) );
339     if ( is_plusererrors_size != 0 ) {
340         plusererror = is_plusererrors[ index( indexX, indexY, indexZ ) ];
341         minusererror = is_minusererrors[ index( indexX, indexY, indexZ ) ];
342     } else {
343         double half = is_errors[ index( indexX, indexY, indexZ ) ] * 0.5;
344         plusererror = half;
345         minusererror = half;
346     }
347 }

```

5.10.2.9 void OHHistogramData3D::reset_bins (void) [virtual]

Reset the bin content.

Reset the bin content, this function must be called before the error type associated to the bins is changed.

See also:

`set_error()` (p. 66) and `set_perror()` (p. 69)

Implements **OHHistogramData** (p. 35).

Definition at line 420 of file `OHHistogramData3D.cxx`.

References `ISHistogram3D::is_bins`, `ISHistogram3D::is_bins_size`, `ISHistogram3D::is_errors`, `ISHistogram3D::is_errors_size`, `ISHistogram3D::is_minusererrors`, `ISHistogram3D::is_minusererrors_size`, `ISHistogram3D::is_plusererrors`, and `ISHistogram3D::is_plusererrors_size`.

Referenced by `set_partition`.

```

420                                     {
421     if ( is_bins_size != 0 ) delete[] is_bins;
422     if ( is_errors_size != 0 ) delete[] is_errors;
423     if ( is_plusererrors_size != 0 ) delete[] is_plusererrors;
424     if ( is_minusererrors_size != 0 ) delete[] is_minusererrors;
425     is_bins_size = is_errors_size = is_plusererrors_size =
426     is_minusererrors_size = 0;
427 }
```

5.10.2.10 `void OHHistogramData3D::set_annotations (const vector< string > & labels, const vector< string > & values)` [virtual]

Update annotations.

The argument vectors must have the same length.

Implements **OHHistogramData** (p. 35).

Definition at line 27 of file `OHHistogramData3D.cxx`.

References `ISHistogram3D::is_annotations`, and `ISHistogram3D::is_annotations_size`.

```

28                                     {
29     assert( labels.size( ) == values.size( ) );
30     size_t size = labels.size( ) * 2;
31     if ( is_annotations_size != size ) {
32         if ( is_annotations_size != 0 ) delete[] is_annotations;
33         is_annotations_size = size;
34         is_annotations = new string[ is_annotations_size ];
35     }
36     for ( size_t i = 0; i < labels.size( ); i++ ) {
37         is_annotations[ i*2 ] = labels[ i ];
38         is_annotations[ i*2 + 1 ] = values[ i ];
39     }
40 }
```

5.10.2.11 `void OHHistogramData3D::set_error (const double & error, long indexX, long indexY, long indexZ)` [virtual]

Update the error associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 61) + 1 or Overflow refer to the overflow bin.

This function may not be used with `set_perror()` (p. 69), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by `error_type()` (p. 60) to `SINGLE_ERROR`.

See also:

`reset_bins()` (p. 65)

Implements **OHHistogramData** (p. 35).

Definition at line 314 of file OHHistogramData3D.cxx.

References `ISHistogram3D::is_errors`, `ISHistogram3D::is_minusererrors_size`, and `ISHistogram3D::is_plusererrors_size`.

```

317                                     {
318     assert( is_plusererrors_size == 0 && is_minusererrors_size == 0 );
319     assert( error >= 0 );
320     alloc_errors( );
321     is_errors[ index( indexX, indexY, indexZ ) ] = error;
322 }
```

5.10.2.12 void OHHistogramData3D::set_height (const double & height, long indexX, long indexY, long indexZ) [virtual]

Update the height (value) of a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index `bin_count()` (p. 61) + 1 or Overflow refer to the overflow bin.

Implements **OHHistogramData** (p. 36).

Definition at line 265 of file OHHistogramData3D.cxx.

References `ISHistogram3D::is_bins`.

```

268                                     {
269     alloc_heights( );
270     is_bins[ index( indexX, indexY, indexZ ) ] = height;
271 }
```

5.10.2.13 void OHHistogramData3D::set_partition (const double & offset, const double & binwidth, long bins, Axis axis) [virtual]

Create a fixed partition for an axis.

This function invalidates bin contents.

Parameters:

offset Lower edge for smallest bin.

binwidth The fixed width of a bin.

bins Number of bins.

Implements **OHHistogramData** (p. 36).

Definition at line 218 of file OHHistogramData3D.cxx.

References `ISHistogram3D::is_xaxis`, `ISHistogram3D::is_xaxis_size`, `ISHistogram3D::is_xtype`, `ISHistogram3D::is_yaxis`, `ISHistogram3D::is_yaxis_size`, `ISHistogram3D::is_ytype`, `ISHistogram3D::is_zaxis`, `ISHistogram3D::is_zaxis_size`, `ISHistogram3D::is_ztype`, and `reset_bins`.

```

221                                     {
222     assert( bins > 0 );
223     assert( binwidth > 0 );
224     reset_bins( );
225     double **ax;
226     size_t *sz;
227     if ( axis == X ) {
228         ax = &is_xaxis;
229         sz = &is_xaxis_size;
230         is_xtype = IS_FIXED;
231     } else if ( axis == Y ) {
232         ax = &is_yaxis;
233         sz = &is_yaxis_size;
234         is_ytype = IS_FIXED;
235     } else {
236         ax = &is_zaxis;
237         sz = &is_zaxis_size;
238         is_ztype = IS_FIXED;
239     }
240     if ( *sz != 3 ) {
241         if ( *sz != 0 ) delete[] *ax;
242         *sz = 3;
243         *ax = new double[ *sz ];
244     }
245     (*ax)[ 0 ] = offset;
246     (*ax)[ 1 ] = binwidth;
247     (*ax)[ 2 ] = bins;
248 }

```

5.10.2.14 void OHHistogramData3D::set_partition (const vector< double > & *partition*, Axis *axis*) [virtual]

Create a variable partition for an axis.

This function invalidates bin contents.

Parameters:

partition A double vector of bin edges.

Implements **OHHistogramData** (p. 36).

Definition at line 183 of file OHHistogramData3D.cxx.

References ISHistogram3D::is_xaxis, ISHistogram3D::is_xaxis_size, ISHistogram3D::is_xtype, ISHistogram3D::is_yaxis, ISHistogram3D::is_yaxis_size, ISHistogram3D::is_ytype, ISHistogram3D::is_zaxis, ISHistogram3D::is_zaxis_size, ISHistogram3D::is_ztype, and reset_bins.

```

184                                     {
185     assert( partition.size( ) > 1 );
186     reset_bins( );
187     double **ax;
188     size_t *sz;
189     if ( axis == X ) {
190         ax = &is_xaxis;
191         sz = &is_xaxis_size;
192         is_xtype = IS_VARIABLE;
193     } else if ( axis == Y ) {
194         ax = &is_yaxis;
195         sz = &is_yaxis_size;
196         is_ytype = IS_VARIABLE;
197     } else {

```

```

198     ax = &is_zaxis;
199     sz = &is_zaxis_size;
200     is_ztype = IS_VARIABLE;
201 }
202 if ( *sz != partition.size( ) ) {
203     if ( *sz != 0 ) delete[] *ax;
204     *sz = partition.size( );
205     *ax = new double[ *sz ];
206 }
207 for ( size_t i = 0; i < *sz; i++ ) {
208     (*ax)[ i ] = partition[ i ];
209 }
210 }

```

5.10.2.15 void OHHistogramData3D::set_perror (const double & *pluserror*, const double & *minuserror*, long *indexX*, long *indexY*, long *indexZ*) [virtual]

Update the plus and minus errors associated to a bin.

Bin indices start at 1, index 0 or Underflow refer to the underflow bin and index **bin_count()** (p. 61) + 1 or Overflow refer to the overflow bin.

This function may not be used with **set_error()** (p. 66), only one of the two types of errors is allowed for one and the same histogram.

This function will set the error type returned by **error_type()** (p. 60) to PLUSMINUS_ERROR.

See also:

reset_bins() (p. 65)

Implements **OHHistogramData** (p. 36).

Definition at line 361 of file OHHistogramData3D.cxx.

References **ISHistogram3D::is_errors_size**, **ISHistogram3D::is_minuserrors**, and **ISHistogram3D::is_pluserrors**.

```

365                                     {
366     assert( is_errors_size == 0 );
367     assert( pluserror >= 0 && minuserror >= 0 );
368     alloc_perrors( );
369     is_pluserrors[ index( indexX, indexY, indexZ ) ] = pluserror;
370     is_minuserrors[ index( indexX, indexY, indexZ ) ] = minuserror;
371 }

```

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

- **OHHistogramData3D.h**
- **OHHistogramData3D.inl**
- **OHHistogramData3D.cxx**

5.11 OHHistogramIterator Class Reference

Histogram Iterator.

```
#include <OHHistogramIterator.h>
```

Public Types

- enum **Months** { **JAN** = 1, **FEB**, **MAR**, **APR**, **MAY**, **JUN**, **JUL**, **AUG**, **SEP**, **OKT**, **NOV**, **DEC** }

Public Methods

- **OHHistogramIterator** (IPCPartition &p, const string &server, const string &provider=".*", const string &histoname=".*", long year=ANY_YEAR, long month=ANY_MONTH, long day=ANY_DAY, long hour=ANY_HOUR, long minute=ANY_MINUTE, long second=ANY_SECOND)
Create a new histogram iterator.
- virtual ~**OHHistogramIterator** ()
Destructor.
- **operator bool** () const
Validity test.
- **bool valid** (void) const
Validity test.
- **bool operator++** ()
Advance the iterator one position.
- **bool operator++** (int)
Advance the iterator one position.
- **bool operator--** ()
Retreat the iterator one position.
- **bool operator--** (int)
Retreat the iterator one position.
- **bool retrieve** (**OHHistogramReceiver** &receiver)
Retrieve the current histogram or set of histograms.
- **string provider** (void) const
Retrieve name of provider who published the current histogram.
- **string name** (void) const
Retrieve name of the current histogram.
- **OWLTime time** () const
Retrieve time when the histogram was published.

Static Public Attributes

- const long **ANY_YEAR** = 0
- const long **ANY_MONTH** = 0
- const long **ANY_DAY** = 0
- const long **ANY_HOUR** = -1
- const long **ANY_MINUTE** = -1
- const long **ANY_SECOND** = -1

5.11.1 Detailed Description

Histogram Iterator.

This class provides functionality to select, list and retrieve histograms from the OH.

See also:

OHHistogramSubscriber (p. 88)

Definition at line 23 of file OHHistogramIterator.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 OHHistogramIterator::OHHistogramIterator (IPCPartition & *p*, const string & *server*, const string & *provider* = ".*", const string & *histoname* = ".*", long *year* = ANY_YEAR, long *month* = ANY_MONTH, long *day* = ANY_DAY, long *hour* = ANY_HOUR, long *minute* = ANY_MINUTE, long *second* = ANY_SECOND)

Create a new histogram iterator.

Immediately after construction the "current histogram" is undefined and the iterator must be advanced with one of the ++ operators.

There is a restriction to the selection of histograms in time, it is not possible to specify a more restrictive attribute (e.g. a day) if the more unrestrictive attributes isn't specified (e.g. the month and the year).

Parameters:

p A valid IPCPartition

server Name of a valid OH server

provider Optional name of histogram provider, should only contain [a-z], [A-Z], [0-9], '_' and optional wildcars according to the Rogue Wave style of regular expressions (see the RWCTRegexp class in the Rogue Wave library)

histoname Optional name of histogram, should only contain [a-z], [A-Z], [0-9], '_' and optional wildcars according to the Rogue Wave style of regular expressions (see the RWCTRegexp class in the Rogue Wave library)

year Optional year when the histogram was published

month Optional month when the histogram was published (JAN-DEC)

day Optional day when the histogram was published (1-31)

hour Optional hour when the histogram was published (0-23)

minute Optional minute when the histogram was published (0-59)

second Optional second when the histogram was published (0-59)

Definition at line 39 of file OHHistogramIterator.cxx.

References provider.

```

48                                     :
49                                     valid_( false ),
50                                     iterator_( 0 )
51 {
52     // Print debug info to cout
53     #ifdef DEBUG
54     std::cout << "Creating OHHistogramIterator for server " << server <<
55         " in partition " << p.name( ) << "." << std::endl;
56     #endif // DEBUG
57
58     // Check arguments
59     if ( ! ( server.size( ) > 0 &&
60         provider.size( ) > 0 &&
61         histoname.size( ) > 0 &&
62         provider.find( ":" ) == string::npos &&
63         histoname.find( ":" ) == string::npos &&
64         ( year > 1900 || year == ANY_YEAR ) &&
65         ( ( month >= JAN && month <= DEC ) || month == ANY_MONTH ) &&
66         ( ( day > 0 && day <= 31 ) || day == ANY_DAY ) &&
67         ( ( hour > 0 && hour < 24 ) || hour == ANY_HOUR ) &&
68         ( ( minute >= 0 && minute < 60 ) || minute == ANY_MINUTE ) &&
69         ( ( second >= 0 && second < 60 ) ) || second == ANY_SECOND ) )
70         return; // Invalidates object
71
72     // Create identifier pattern
73     char buf[16];
74     string id( "OH:" );
75     id += ( provider + ":" + histoname + ":" );
76     if ( year != ANY_YEAR ) {
77         sprintf( buf, "%ld\n", year );
78         id += buf;
79         if ( month != ANY_MONTH ) {
80             sprintf( buf, "%02ld\n", month );
81             id += buf;
82             if ( day != ANY_DAY ) {
83                 sprintf( buf, "%02ld\n", day );
84                 id += buf;
85                 if ( hour != ANY_HOUR ) {
86                     sprintf( buf, "%02ld\n", hour );
87                     id += buf;
88                     if ( minute != ANY_MINUTE ) {
89                         sprintf( buf, "%02ld\n", minute );
90                         id += buf;
91                         if ( second != ANY_SECOND ) {
92                             sprintf( buf, "%02ld\n", second );
93                             id += buf;
94                         }
95                     }
96                 }
97             }
98         }
99     }
100     id += ".*:.*";
101
102     // Print debug info to cout
103     #ifdef DEBUG
104     std::cout << " Selecting histograms with an id matching " <<
105         id << "." << std::endl;

```

```

106     #endif // DEBUG
107
108     // Create IS iterator
109     iterator_ = new ISInfoIterator( p, server.c_str( ), id.c_str( ) );
110
111     // Check if construction was successfully
112     if ( iterator_ != 0 ) valid_ = true;
113 }

```

5.11.3 Member Function Documentation

5.11.3.1 string OHHistogramIterator::name (void) const

Retrieve name of the current histogram.

Returns:

The name of the the histogram (set) on the current position or "" if the current position is undefined.

Definition at line 199 of file OHHistogramIterator.cxx.

Referenced by provider, retrieve, and time.

```

199                                     {
200     assert( *this );
201     const char* n = iterator_ -> name( );
202     if ( n == 0 ) return string( "" );
203     // Need to extract typename from "OH:providername:typename:*"
204     return OHUtils::extract_subid( n, 2 );
205 }

```

5.11.3.2 OHHistogramIterator::operator bool () const

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

[valid\(\)](#) (p. 76)

Definition at line 123 of file OHHistogramIterator.cxx.

```

123                                     {
124     return valid_;
125 }

```

5.11.3.3 bool OHHistogramIterator::operator++ (int)

Advance the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 146 of file OHHistogramIterator.cxx.

```
146                                     {
147     assert( *this );
148     return (*iterator_)+;
149 }
```

5.11.3.4 bool OHHistogramIterator::operator++ ()

Advance the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 138 of file OHHistogramIterator.cxx.

```
138                                     {
139     assert( *this );
140     return (*iterator_)+;
141 }
```

5.11.3.5 bool OHHistogramIterator::operator- (int)

Retreat the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 162 of file OHHistogramIterator.cxx.

```
162                                     {
163     assert( *this );
164     return (*iterator_)-;
165 }
```

5.11.3.6 bool OHHistogramIterator::operator- ()

Retreat the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 154 of file OHHistogramIterator.cxx.

```
154                                     {
155     assert( *this );
156     return (*iterator_)-;
157 }
```

5.11.3.7 string OHHistogramIterator::provider (void) const

Retrieve name of provider who published the current histogram.

Returns:

The name of the provider who published the histogram (set) on the current position or "" if the current position is undefined.

Definition at line 187 of file OHHistogramIterator.cxx.

References name.

Referenced by OHHistogramIterator.

```

187                                     {
188     assert( *this );
189     const char* n = iterator_ -> name( );
190     if ( n == 0 ) return string( "" );
191     // Need to extract provider from "OH:providername:*"
192     return OHUtils::extract_subid( n, 1 );
193 }
```

5.11.3.8 bool OHHistogramIterator::retrieve (OHHistogramReceiver & receiver)

Retrieve the current histogram or set of histograms.

Parameters:

receiver Should be a user defined histogram receiver object derived from one of the OH receiver classes.

Returns:

true if the histogram or histogram set was successfully received by the user (i.e. You!). If a communication error or other OH internal error occurs it may happen that the receiver class is never notified before false is returned.

Definition at line 175 of file OHHistogramIterator.cxx.

References name, and OHHistogramReceiver::receive.

```

175                                     {
176     assert( *this );
177     if ( iterator_ -> name( ) != 0 ) {
178         return receiver.receive( iterator_ );
179     }
180     return false;
181 }
```

5.11.3.9 OWLTime OHHistogramIterator::time () const

Retrieve time when the histogram was published.

Returns:

The time when the histogram (set) on the current position was published or OWLTime() if the current position is undefined.

Definition at line 211 of file OHHistogramIterator.cxx.

References name.

```
211                                     {
212     assert( *this );
213     const char* n = iterator_ -> name( );
214     if ( n == 0 ) return OWLTime( );
215     // Need to extract time from "OH:providername:typename:YYYYMMDDhhmmss:*"
216     string t = OHUtils::extract_subid( n, 3 );
217     t = t.substr( 6, 2 ) + "/" + t.substr( 4, 2 ) + "/" + t.substr( 0, 4 ) +
218         " " + t.substr( 8, 2 ) + ":" + t.substr( 10, 2 ) + ":" +
219         t.substr( 12, 2 );
220     return OWLTime( t.c_str( ) );
221 }
```

5.11.3.10 bool OHHistogramIterator::valid (void) const

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

operator bool() (p. 73)

Definition at line 131 of file OHHistogramIterator.cxx.

```
131                                     {
132     return valid_;
133 }
```

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

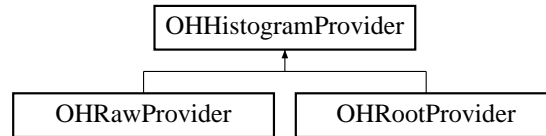
- **OHHistogramIterator.h**
- **OHHistogramIterator.cxx**

5.12 OHHistogramProvider Class Reference

Histogram Provider baseclass.

```
#include <OHHistogramProvider.h>
```

Inheritance diagram for OHHistogramProvider::



Public Methods

- **OHHistogramProvider** (const IPCPartition &p, const string &server, const string &name)
Create a new histogram provider.
- virtual **~OHHistogramProvider** ()
Destructor.
- virtual **operator bool** ()
Validity test.
- virtual bool **valid** (void)
Validity test.

Protected Types

- enum **Status** { **Success**, **CommFailure**, **InvalidArgument** }

Protected Methods

- Status **publish** (const **OHHistogramData1D** &histogram, const string &name, bool update=false)
Publish a one dimensional histogram.
- Status **publish** (const **OHHistogramData2D** &histogram, const string &name, bool update=false)
Publish a two dimensional histogram.
- Status **publish** (const **OHHistogramData3D** &histogram, const string &name, bool update=false)
Publish a three dimensional histogram.

5.12.1 Detailed Description

Histogram Provider baseclass.

This class provides functionality to publish the OH internal histogram types and thus acts as an abstraction layer to the underlying technology used to transport and buffer histograms . Derive a new class for each external package (or histogram format) which should export histograms to the OH.

Symbolic example:

```
class MyProvider : public OHHistogramProvider {
public:
    MyProvider( const IPCPartition & p,
               const string & server,
               const string & name ) :
        OHHistogramProvider( p, server, name ) {
    }

    Status publish( const MyHistogram1D& h, const string & name, bool update ) {
        OHHistogramData1D data;
        // Copy content from h to data
        data.set_title( h.title( ) );

        // More conversion stuff goes here...

        // Publish the histogram
        return OHHistogramProvider::publish( data, name, true );
    }

    // Here some functions for other histogram types
    // could be added.
};
```

For more information on how to convert to the OH internal histogram representations see **OHHistogramData1D** (p.38), **OHHistogramData2D** (p.48) and **OHHistogramData3D** (p.59).

See also:

OHRootProvider (p.108) and **OHRawProvider** (p.97)

Definition at line 63 of file OHHistogramProvider.h.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 OHHistogramProvider::OHHistogramProvider (const IPCPartition & p, const string & server, const string & name)

Create a new histogram provider.

Parameters:

p A valid IPCPartition

server Name of a valid OH server (An IS server in partition p)

name Name of the provider, user is responsible for specifying a unique name. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

Definition at line 31 of file OHHistogramProvider.cxx.

```

33                                     :
34         valid_( false ),
35         partition_( p ),
36         server_( server ),
37         provider_( name ),
38         dictionary_( p ) {
39     // Print debug info to cout
40     #ifdef DEBUG
41         std::cout << "Creating OHHistogramProvider " << name.c_str() <<
42             " on server " << server_.c_str() << " in partition " <<
43             p.name() << "." << std::endl;
44     #endif // DEBUG
45
46     // Check arguments
47     if ( ! ( name.size() > 0 &&
48         server_.size() > 0 &&
49         name.find( ":" ) == string::npos ) ) return; // Invalidates object
50
51     // Add the provider name to the IS server, success will imply
52     // that the provider has a unique name within this server. This
53     // information is also used by the OHProviderIterator class to
54     // retrieve active providers.
55     ISInfo::Status s;
56     ISInfoString providerinfo( provider_ );
57     s = dictionary_.insert( ( server_ + ".OHP:" + provider_ ).c_str(),
58         providerinfo );
59
60     // Check if construction was successful
61     if ( s == ISInfo::Success )
62         valid_ = true;
63 }
```

5.12.3 Member Function Documentation

5.12.3.1 OHHistogramProvider::operator bool () [virtual]

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

[valid\(\)](#) (p. 83)

Reimplemented in [OHRawProvider](#) (p. 98).

Definition at line 73 of file OHHistogramProvider.cxx.

```

73         {
74     return valid_;
75 }
```

5.12.3.2 OHHistogramProvider::Status OHHistogramProvider::publish (const OHHistogramData3D & *histogram*, const string & *name*, bool *update* = false) [protected]

Publish a three dimensional histogram.

Parameters:

histogram A valid non empty histogram

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

update A flag which decides about histogram update (by default false)

Returns:

CommFailure if a communication error occurred, InvalidArgument if one or more arguments is invalid else Success.

Definition at line 224 of file OHHistogramProvider.cxx.

```

225                                     {
226     assert( *this );
227     ISInfo::Status s;
228     long counter = 0;
229     if ( name.find( ":" ) != string::npos ) return InvalidArgument;
230     if (update)
231     {
232         bool find=false;
233         ISInfoIterator is( partition_, server_.c_str(),".*");
234         while ( is() )
235         {
236             string tmpName = OHUtils::extract_subid( is.name(), 2 );
237             string tmpTime = OHUtils::extract_subid( is.name(), 3 );
238             string tmpUpdate = OHUtils::extract_subid( is.name(), 4 );
239             IStype ist=is.type();
240             if (tmpName==name && tmpUpdate=="1" && ist.name()=="ISHistogram3D")
241             {
242                 find=true;
243                 string id = create_name( name, true, tmpTime );
244                 string id1 = create_name( name, true );
245                 do {
246                     string hid = create_id( id, counter++ );
247                     s = dictionary_.remove( hid.c_str( ) );
248                 } while ( s == ISInfo::AlreadyExist );
249                 counter=0;
250                 do {
251                     string hid1 = create_id( id1, counter++ );
252                     s = dictionary_.insert( hid1.c_str( ), const_cast<OHHistogramData3D*>( histogram ) );
253                 } while ( s == ISInfo::AlreadyExist );
254                 if ( s == ISInfo::Success ) return Success;
255             }
256         }
257     if (!find)
258     {
259         string id = create_name( name, true );
260         do {
261             string hid = create_id( id, counter++ );
262             s = dictionary_.insert( hid.c_str( ), const_cast<OHHistogramData3D*>( histogram ) );
263         } while ( s == ISInfo::AlreadyExist );
264         if ( s == ISInfo::Success ) return Success;
265     }
266 }

```

```

267     else
268     {
269         string id = create_name( name );
270         do
271         {
272             string hid = create_id( id, counter++ );
273             s = dictionary_.insert( hid.c_str( ), const_cast<OHHistogramData3D*>( histogram ) );
274         } while ( s == ISInfo::AlreadyExist );
275         if ( s == ISInfo::Success ) return Success;
276     }
277     return CommFailure;
278 }

```

5.12.3.3 OHHistogramProvider::Status OHHistogramProvider::publish (const OHHistogramData2D & *histogram*, const string & *name*, bool *update* = false) [protected]

Publish a two dimensional histogram.

Parameters:

histogram A valid non empty histogram

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

update A flag which decides about histogram update (by default false)

Returns:

CommFailure if a communication error occurred, InvalidArgument if one or more arguments is invalid else Success.

Definition at line 159 of file OHHistogramProvider.cxx.

```

160                                     {
161     assert( *this );
162     ISInfo::Status s;
163     long counter = 0;
164     if ( name.find( ":" ) != string::npos ) return InvalidArgument;
165     if (update)
166     {
167         bool find=false;
168         ISInfoIterator is( partition_, server_.c_str(), ".*");
169         while ( is() )
170         {
171             string tmpName = OHUtils::extract_subid( is.name(), 2 );
172             string tmpTime = OHUtils::extract_subid( is.name(), 3 );
173             string tmpUpdate = OHUtils::extract_subid( is.name(), 4 );
174             IStype ist=is.type();
175             if (tmpName==name && tmpUpdate=="1" && ist.name()=="ISHistogram2D")
176             {
177                 find=true;
178                 string id = create_name( name, true, tmpTime );
179                 string id1 = create_name( name, true );
180                 do {
181                     string hid = create_id( id, counter++ );
182                     s = dictionary_.remove( hid.c_str( ));
183                 } while ( s == ISInfo::AlreadyExist );
184                 counter=0;
185                 do {
186                     string hid1 = create_id( id1, counter++ );

```

```

187         s = dictionary_.insert( hid1.c_str( ), const_cast<OHHistogramData2D*>( histogram ) );
188     } while ( s == ISInfo::AlreadyExist );
189     if ( s == ISInfo::Success ) return Success;
190 }
191 }
192 if (!find)
193 {
194     string id = create_name( name, true );
195     do {
196         string hid = create_id( id, counter++ );
197         s = dictionary_.insert( hid.c_str( ), const_cast<OHHistogramData2D*>( histogram ) );
198     } while ( s == ISInfo::AlreadyExist );
199     if ( s == ISInfo::Success ) return Success;
200 }
201 }
202 else
203 {
204     string id = create_name( name );
205     do
206     {
207         string hid = create_id( id, counter++ );
208         s = dictionary_.insert( hid.c_str( ), const_cast<OHHistogramData2D*>( histogram ) );
209     } while ( s == ISInfo::AlreadyExist );
210     if ( s == ISInfo::Success ) return Success;
211 }
212 return CommFailure;
213 }

```

5.12.3.4 OHHistogramProvider::Status OHHistogramProvider::publish (const OHHistogramData1D & *histogram*, const string & *name*, bool *update* = false) [protected]

Publish a one dimensional histogram.

Parameters:

histogram A valid non empty histogram

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

update A flag which decides about histogram update (by default false)

Returns:

CommFailure if a communication error occurred, InvalidArgument if one or more arguments is invalid else Success.

Definition at line 94 of file OHHistogramProvider.cxx.

Referenced by OHRootProvider::publish, and OHRawProvider::publish.

```

95                                     {
96     assert( *this );
97     ISInfo::Status s;
98     long counter = 0;
99     if ( name.find( ":" ) != string::npos ) return InvalidArgument;
100     if (update)
101     {
102         bool find=false;
103         ISInfoIterator is( partition_, server_.c_str(), ".*");
104         while ( is()

```

```

105     {
106         string tmpName = OHUtils::extract_subid( is.name(), 2 );
107         string tmpTime = OHUtils::extract_subid( is.name(), 3 );
108         string tmpUpdate = OHUtils::extract_subid( is.name(), 4 );
109         IStype ist=is.type();
110         if (tmpName==name && tmpUpdate=="1" && ist.name()=="ISHistogramID")
111             {
112                 find=true;
113                 string id = create_name( name, true, tmpTime );
114                 string id1 = create_name( name, true );
115                 do {
116                     string hid = create_id( id, counter++ );
117                     s = dictionary_.remove( hid.c_str( ) );
118                 } while ( s == ISInfo::AlreadyExist );
119                 counter=0;
120                 do {
121                     string hid1 = create_id( id1, counter++ );
122                     s = dictionary_.insert( hid1.c_str( ), const_cast<OHHistogramData1D&>( histogram ) );
123                 } while ( s == ISInfo::AlreadyExist );
124                 if ( s == ISInfo::Success ) return Success;
125             }
126     }
127     if (!find)
128     {
129         string id = create_name( name, true );
130         do {
131             string hid = create_id( id, counter++ );
132             s = dictionary_.insert( hid.c_str( ), const_cast<OHHistogramData1D&>( histogram ) );
133         } while ( s == ISInfo::AlreadyExist );
134         if ( s == ISInfo::Success ) return Success;
135     }
136 }
137 else
138 {
139     string id = create_name( name );
140     do
141     {
142         string hid = create_id( id, counter++ );
143         s = dictionary_.insert( hid.c_str( ), const_cast<OHHistogramData1D&>( histogram ) );
144     } while ( s == ISInfo::AlreadyExist );
145     if ( s == ISInfo::Success ) return Success;
146 }
147 return CommFailure;
148 }

```

5.12.3.5 bool OHHistogramProvider::valid (void) [virtual]

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

operator bool() (p. 79)

Reimplemented in **OHRawProvider** (p. 105).

Definition at line 81 of file OHHistogramProvider.cxx.

Referenced by **OHRootProvider::operator bool**, **OHRawProvider::operator bool**, **OHRootProvider::valid**, and **OHRawProvider::valid**.

```
81                                     {  
82     return valid_;  
83 }
```

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

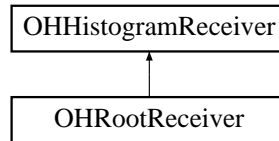
- **OHHistogramProvider.h**
- **OHHistogramProvider.cxx**

5.13 OHHistogramReceiver Class Reference

Histogram Receiver baseclass.

```
#include <OHHistogramReceiver.h>
```

Inheritance diagram for OHHistogramReceiver::



Public Methods

- virtual `~OHHistogramReceiver ()`
Destructor.
- virtual `operator bool ()`
Validity test.
- virtual `bool valid (void)`
Validity test.

Protected Methods

- virtual `bool receive_ (OHHistogramData1D *histogram)=0`
Called when a one dimensional histogram is received.
- virtual `bool receive_ (OHHistogramData2D *histogram)=0`
Called when a two dimensional histogram is received.
- virtual `bool receive_ (OHHistogramData3D *histogram)=0`
Called when a three dimensional histogram is received.

Friends

- class `OHHistogramIterator`
- class `OHHistogramSubscriber`

5.13.1 Detailed Description

Histogram Receiver baseclass.

This class provides functionality to receive histograms from an `OHHistogramIterator` (p. 70) or `OHHistogramSubscriber` (p. 88) in the OH internal formats and thus acts as an abstraction

layer to the underlying technology used to transport and buffer histograms. Derive a new class for each external package (or histogram format) which should import histograms from the OH.

Example:

```
class MyReceiver : public OHHistogramReceiver {
protected:

    // Histogram user tasks should override this function to
    // receive histograms
    virtual bool receive( MyHistogram1D* histo ) {
        delete histo;
        return false; // User did not receive the histogram
    }

private:

    bool receive_( OHHistogramData1D* histogram ) {
        // Need to create a new histogram and fill it with info
        // from the received histogram
        MyHistogram1D* h = MyHistogram1D( ... );

        // Conversion stuff goes here...

        // Let the user receive the histogram
        return receive( h );
    }

};
```

For more information on how to convert from the OH internal histogram representations see **OHHistogramData1D** (p.38), **OHHistogramData2D** (p.48) and **OHHistogramData3D** (p.59).

Observe the possibility here to use multiple inheritance in order to access protected members in the non OH histograms.

See also:

OHRootReceiver (p.112)

Definition at line 67 of file OHHistogramReceiver.h.

5.13.2 Member Function Documentation

5.13.2.1 OHHistogramReceiver::operator bool () [inline, virtual]

Validity test.

Returns:

true, objects of this type is valid by default unless derived classes override this function

See also:

valid() (p.87)

Definition at line 163 of file OHHistogramReceiver.h.

```
163         {
164     return true;
165 }
```

5.13.2.2 virtual bool OHHistogramReceiver::receive_ (OHHistogramData3D * *histogram*) [protected, pure virtual]

Called when a three dimensional histogram is received.

Parameters:

histogram A histogram, user is responsible for deleting the histogram after usage.

Returns:

User should return true if the histogram was successfully received.

5.13.2.3 virtual bool OHHistogramReceiver::receive_ (OHHistogramData2D * *histogram*) [protected, pure virtual]

Called when a two dimensional histogram is received.

Parameters:

histogram A histogram, user is responsible for deleting the histogram after usage.

Returns:

User should return true if the histogram was successfully received.

5.13.2.4 virtual bool OHHistogramReceiver::receive_ (OHHistogramData1D * *histogram*) [protected, pure virtual]

Called when a one dimensional histogram is received.

Parameters:

histogram A histogram, user is responsible for deleting the histogram after usage.

Returns:

User should return true if the histogram was successfully received.

5.13.2.5 bool OHHistogramReceiver::valid (void) [inline, virtual]

Validity test.

Returns:

true, objects of this type is valid by default unless derived classes override this function

See also:

operator bool() (p. 86)

Definition at line 173 of file OHHistogramReceiver.h.

```
173         {  
174     return true;  
175 }
```

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

- **OHHistogramReceiver.h**
- **OHHistogramReceiver.cxx**

5.14 OHHistogramSubscriber Class Reference

Histogram subscriber.

```
#include <OHHistogramSubscriber.h>
```

Public Types

- enum **Months** { **JAN** = 1, **FEB**, **MAR**, **APR**, **MAY**, **JUN**, **JUL**, **AUG**, **SEP**, **OKT**, **NOV**, **DEC** }

Public Methods

- **OHHistogramSubscriber** (**OHHistogramReceiver** &receiver, IPCPartition &p, const string &server, const string &provider=".*", const string &histoname=".*", long year=ANY_YEAR, long month=ANY_MONTH, long day=ANY_DAY, long hour=ANY_HOUR, long minute=ANY_MINUTE, long second=ANY_SECOND)

Create a new histogram subscriber.

- operator **bool** ()

Validity test.

- **bool valid** (void)

Validity test.

- **void start** (void)

Start subscription.

- **void stop** (void)

Stop subscription.

Static Public Attributes

- const long **ANY_YEAR** = 0
- const long **ANY_MONTH** = 0
- const long **ANY_DAY** = 0
- const long **ANY_HOUR** = -1
- const long **ANY_MINUTE** = -1
- const long **ANY_SECOND** = -1

Friends

- **void oh_subscriber_is_callback** (ISCallbackInfo *isc)

5.14.1 Detailed Description

Histogram subscriber.

This class provides functionality to subscribe for histograms with specific attributes.

See also:

[OHHistogramIterator](#) (p. 70)

Definition at line 19 of file OHHistogramSubscriber.h.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 OHHistogramSubscriber::OHHistogramSubscriber (OHHistogramReceiver & receiver, IPCPartition & p, const string & server, const string & provider = ".*", const string & histoname = ".*", long year = ANY_YEAR, long month = ANY_MONTH, long day = ANY_DAY, long hour = ANY_HOUR, long minute = ANY_MINUTE, long second = ANY_SECOND)

Create a new histogram subscriber.

There is a restriction to the selection of histograms in time, it is not possible to specify a more restrictive attribute (e.g. a day) if the more unrestrictive attributes isn't specified (e.g. the month and the year).

It is not allowed to subscribe more than once for each combination of the selection parameters.

Parameters:

receiver Should be a user defined histogram receiver object derived from one of the OH receiver classes.

p A valid IPCPartition

server Name of a valid OH server

provider Optional name of histogram provider, should only contain [a-z], [A-Z], [0-9], '_' and optional wildcars according to the Rogue Wave style of regular expressions (see the RWCRexp class in the Rogue Wave library)

histoname Optional name of histogram, should only contain [a-z], [A-Z], [0-9], '_' and optional wildcars according to the Rogue Wave style of regular expressions (see the RWCRexp class in the Rogue Wave library)

year Optional year when the histogram was published

month Optional month when the histogram was published (JAN-DEC)

day Optional day when the histogram was published (1-31)

hour Optional hour when the histogram was published (0-23)

minute Optional minute when the histogram was published (0-59)

second Optional second when the histogram was published (0-59)

Definition at line 49 of file OHHistogramSubscriber.cxx.

```

59                                     :
60                                     valid_( false ),
61                                     isreceiver_( p ),
62                                     receiver_( receiver ) {

```

```

63 // Print debug info to cout
64 #ifdef DEBUG
65 std::cout << "Creating OHHistogramIterator for server " << server <<
66     " in partition " << p.name( ) << "." << std::endl;
67 #endif // DEBUG
68
69 // Check arguments
70 if ( ! ( server.size( ) > 0 &&
71     provider.size( ) > 0 &&
72     histoname.size( ) > 0 &&
73     provider.find( ":" ) == string::npos &&
74     histoname.find( ":" ) == string::npos &&
75     ( year > 1900 || year == ANY_YEAR ) &&
76     ( ( month >= JAN && month <= DEC ) || month == ANY_MONTH ) &&
77     ( ( day > 0 && day <= 31 ) || day == ANY_DAY ) &&
78     ( ( hour > 0 && hour < 24 ) || hour == ANY_HOUR ) &&
79     ( ( minute >= 0 && minute < 60 ) || minute == ANY_MINUTE ) &&
80     ( ( second >= 0 && second < 60 ) || second == ANY_SECOND ) )
81     return; // Invalidates object
82
83 // Create identifier pattern
84 char buf[16];
85 string id( "OH:" );
86 id += ( provider + ":" + histoname + ":" );
87 if ( year != ANY_YEAR ) {
88     sprintf( buf, "%ld\n", year );
89     id += buf;
90     if ( month != ANY_MONTH ) {
91         sprintf( buf, "%02ld\n", month );
92         id += buf;
93         if ( day != ANY_DAY ) {
94             sprintf( buf, "%02ld\n", day );
95             id += buf;
96             if ( hour != ANY_HOUR ) {
97                 sprintf( buf, "%02ld\n", hour );
98                 id += buf;
99                 if ( minute != ANY_MINUTE ) {
100                     sprintf( buf, "%02ld\n", minute );
101                     id += buf;
102                     if ( second != ANY_SECOND ) {
103                         sprintf( buf, "%02ld\n", second );
104                         id += buf;
105                     }
106                 }
107             }
108         }
109     }
110 }
111 id += ".*:.*";
112
113 // Print debug info to cout
114 #ifdef DEBUG
115 std::cout << " Subscribing for histograms with an id matching " << id << "." << std::endl;
116 #endif // DEBUG
117
118 // Register subscription in server
119 ISInfo::Status s;
120 s = isreceiver_ . subscribe( server.c_str( ),
121     id.c_str( ),
122     oh_subscriber_is_callback,
123     this );
124
125 // Check if construction was successfully
126 if ( s == ISInfo::Success ) valid_ = true;
127

```

128 }

5.14.3 Member Function Documentation

5.14.3.1 OHHistogramSubscriber::operator bool ()

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

`valid()` (p.92)

Definition at line 134 of file OHHistogramSubscriber.cxx.

```
134                                     {
135     return valid_;
136 }
```

5.14.3.2 void OHHistogramSubscriber::start (void)

Start subscription.

This function will not return until `stop()` (p.91) is called.

Definition at line 149 of file OHHistogramSubscriber.cxx.

```
149                                     {
150     assert( *this );
151     isreceiver_ . run( );
152 }
```

5.14.3.3 void OHHistogramSubscriber::stop (void)

Stop subscription.

See also:

`run()`

Definition at line 157 of file OHHistogramSubscriber.cxx.

```
157                                     {
158     assert( *this );
159     isreceiver_ . stop( );
160 }
```

5.14.3.4 bool OHHistogramSubscriber::valid (void)

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

operator bool() (p. 91)

Definition at line 142 of file OHHistogramSubscriber.cxx.

```
142                                     {  
143     return valid_;  
144 }
```

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

- OHHistogramSubscriber.h
- OHHistogramSubscriber.cxx

5.15 OHPProviderIterator Class Reference

Provider iterator.

```
#include <OHPProviderIterator.h>
```

Public Methods

- **OHPProviderIterator** (IPCPartition &p, const string &server)
Create a new provider iterator.
- **operator bool** ()
Validity test.
- **bool valid** (void)
Validity test.
- **bool operator++** ()
Advance the iterator one position.
- **bool operator++** (int)
Advance the iterator one position.
- **bool operator--** ()
Retreat the iterator one position.
- **bool operator--** (int)
Retreat the iterator one position.
- **operator string** ()
Retrieve name of current provider.
- **string name** (void)
Retrieve name of current provider.
- **void reset** (void)
Reset the iterator to its initial state.

5.15.1 Detailed Description

Provider iterator.

This class provides functionality to retrieve the names of all currently active histogram providers which publish histograms in a specific OH server. Observe that histograms may exist in the server which have been published by a provider which is not accessible with this iterator.

Definition at line 23 of file OHPProviderIterator.h.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 OHPProviderIterator::OHPProviderIterator (IPCPartition & *p*, const string & *server*)

Create a new provider iterator.

Immediately after construction the "current provider" is undefined and the iterator must be advanced with one of the ++ operators.

Parameters:

- p* A valid IPCPartition
- server* Name of a valid OH server

Definition at line 22 of file OHPProviderIterator.cxx.

```

23                                     :
24         iterator_( p,
25                   server.c_str( ),
26                   searchpattern_ ) {
27 }
```

5.15.3 Member Function Documentation

5.15.3.1 string OHPProviderIterator::name (void)

Retrieve name of current provider.

Returns:

- The name of the current provider or "" if the current provider is undefined.

Definition at line 89 of file OHPProviderIterator.cxx.

```

89                                     {
90     ISInfoString s;
91     if ( iterator_.value( s ) == ISInfo::Success ) {
92         return s;
93     }
94     return string("");
95 }
```

5.15.3.2 OHPProviderIterator::operator bool ()

Validity test.

Returns:

- true if the object was constructed successfully, otherwise false

See also:

- `valid()` (p. 96)

Definition at line 33 of file OHPProviderIterator.cxx.

```
33         {
34     return true;
35 }
```

5.15.3.3 OHProviderIterator::operator string ()

Retrieve name of current provider.

Returns:

The name of the current provider or "" if the current provider is undefined.

Definition at line 77 of file OHProviderIterator.cxx.

```
77         {
78     ISInfoString s;
79     if ( iterator_.value( s ) == ISInfo::Success ) {
80         return s;
81     }
82     return string("");
83 }
```

5.15.3.4 bool OHProviderIterator::operator++ (int)

Advance the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 55 of file OHProviderIterator.cxx.

```
55         {
56     return iterator_++;
57 }
```

5.15.3.5 bool OHProviderIterator::operator++ ()

Advance the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 48 of file OHProviderIterator.cxx.

```
48         {
49     return iterator_++;
50 }
```

5.15.3.6 bool OHPProviderIterator::operator-- (int)

Retreat the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 69 of file OHPProviderIterator.cxx.

```
69         {
70     return iterator--;
71 }
```

5.15.3.7 bool OHPProviderIterator::operator-- ()

Retreat the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 62 of file OHPProviderIterator.cxx.

```
62         {
63     return iterator--;
64 }
```

5.15.3.8 bool OHPProviderIterator::valid (void)

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

operator bool() (p. 94)

Definition at line 41 of file OHPProviderIterator.cxx.

```
41         {
42     return true;
43 }
```

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

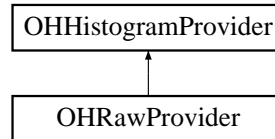
- **OHPProviderIterator.h**
- **OHPProviderIterator.cxx**

5.16 OHRawProvider Class Template Reference

RAW Provider.

```
#include <OHRawProvider.h>
```

Inheritance diagram for OHRawProvider::



Public Methods

- **OHRawProvider** (const IPCPartition &p, const string &server, const string &name)
Create a new Raw provider.
- virtual **operator bool** ()
Validity test.
- virtual **bool valid** (void)
Validity test.
- Status **publish** (const string &name, const string &title, const string &label, long bincount, TAxis offset, TAxis width, TContent *contents, TError *errors, bool outOfRangeBins, const vector< string > &labels, const vector< string > &values)
Publish a 1D histogram with fixed width bins.
- Status **publish** (const string &name, const string &title, const string &label, long bincount, TAxis *axis, TContent *contents, TError *errors, bool outOfRangeBins, const vector< string > &labels, const vector< string > &values)
Publish a 1D histogram with variable width bins.
- Status **publish** (const string &name, const string &title, const string &xlabel, long xcount, TAxis xoffset, TAxis xwidth, const string &ylabel, long ycount, TAxis yoffset, TAxis ywidth, TContent *contents, TError *errors, bool outOfRangeBins, const vector< string > &labels, const vector< string > &values)
Publish a 2D histogram with fixed width bins.
- Status **publish** (const string &name, const string &title, const string &xlabel, long xcount, TAxis *xaxis, const string &ylabel, long ycount, TAxis *yaxis, TContent *contents, TError *errors, bool outOfRangeBins, const vector< string > &labels, const vector< string > &values)
Publish a 2D histogram with variable width bins.

Protected Attributes

- TMap **map_**

5.16.1 Detailed Description

```
template<class TContent, class TError, class TAxis, class TMap = OHRawProvider-
DM> class OHRawProvider< TContent, TError, TAxis, TMap >
```

RAW Provider.

This template provides functionality to export histograms represented by arrays of some fundamental data type to the OH. The meaning of the template parameters are as follows

TContent: The data type used by user to represent bin contents (heights).

TError: The data type used by user to represent bin errors.

TAxis: The data type used by user to represent axis partitions.

TMap: Governs how the user bins are accessed. see **OHRawProviderDM** (p.106) (optional).

See also:

OHRawProviderDM (p.106) , **OHRootProvider** (p.108)

Todo:

Support 3D histograms if necessary.

Definition at line 124 of file OHRawProvider.h.

5.16.2 Constructor & Destructor Documentation

```
5.16.2.1 template<class TContent, class TError, class TAxis, class TMap>
OHRawProvider< TContent, TError, TAxis, TMap >::OHRawProvider
(const IPCPartition & p, const string & server, const string & name)
```

Create a new Raw provider.

Parameters:

p A valid IPCPartition

server Name of a valid OH server

name Name of the provider, user is responsible for specifying a unique name. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

Definition at line 37 of file OHRawProvider.inl.

```
39                                     :
40 OHHistogramProvider( p, server, name ) { }
```

5.16.3 Member Function Documentation

```
5.16.3.1 template<class TContent, class TError, class TAxis, class TMap>
OHRawProvider< TContent, TError, TAxis, TMap >::operator bool ()
[virtual]
```

Validity test.

Returns:

true if the provider was successfully created, otherwise false.

See also:

`valid()` (p. 105)

Reimplemented from `OHHistogramProvider` (p. 79).

Definition at line 47 of file `OHRawProvider.inl`.

References `OHHistogramProvider::valid`.

```

47                                     {
48     return OHHistogramProvider::valid( );
49 }
```

5.16.3.2 `template<class TContent, class TError, class TAxis, class TMap>
OHRawProvider< TContent, TError, TAxis, TMap >::Status
OHRawProvider< TContent, TError, TAxis, TMap >::publish (const
string & name, const string & title, const string & xlabel, long xcount,
TAxis * xaxis, const string & ylabel, long ycount, TAxis * yaxis, TContent
* contents, TError * errors, bool outOfRangeBins, const vector< string >
& labels, const vector< string > & values)`

Publish a 2D histogram with variable width bins.

Parameters:

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and `.`.

title The histogram title

xlabel X Axis label

xcount The number of bins along the x axis, excluding underflow and overflow

xaxis A pointer to the x axis partition (*xcount* + 1 values)

ylabel Y Axis label

ycount The number of bins along the y axis, excluding underflow and overflow

yaxis A pointer to the y axis partition (*ycount* + 1 values)

contents A pointer to the position where the bin contents (heights) are located.

errors A pointer to the position where the bin errors are located or 0 if no errors.

outOfRangeBins Out of range bins or not

labels Labels for any annotations that should be attached to the histogram.

values Values for any annotations that should be attached to the histogram.

Returns:

`OHRawProvider::CommFailure` if a communication error occurred, `OHRawProvider::InvalidArgument` if one or more arguments is invalid, else `OHRawProvider::Success`.

See also:

`OHRawProviderDM` (p. 106)

Definition at line 340 of file OHRawProvider.inl.

References OHHistogramProvider::publish.

```

352                                     {
353     assert( contents );
354     long i;
355     vector<double> part;
356     OHHistogramData2D* histo = new OHHistogramData2D;
357     histo -> set_title( title );
358     histo -> set_label( xlabel, OHHistogramData2D::X );
359     histo -> set_label( ylabel, OHHistogramData2D::Y );
360     histo -> set_annotations( labels, values );
361     for ( i = 0; i <= xcount; i++ ) part.push_back( xaxis[ i ] );
362     histo -> set_partition( part, OHHistogramData2D::X );
363     part.clear( );
364     for ( i = 0; i <= xcount; i++ ) part.push_back( yaxis[ i ] );
365     histo -> set_partition( part, OHHistogramData2D::Y );
366     part.clear( );
367     long wx = xcount, wy = ycount, off;
368     if ( outOfRangeBins ) {
369         wx += 2;
370         wy += 2;
371         off = 0;
372     } else {
373         off = 1;
374         long i;
375         // Clear underflow and overflow
376         for ( i = 0; i <= wx+1; i++ ) {
377             histo -> set_height( 0, i, 0 );
378             histo -> set_height( 0, i, wy+1 );
379         }
380         for ( i = 1; i <= wy; i++ ) {
381             histo -> set_height( 0, 0, i );
382             histo -> set_height( 0, wx+1, i );
383         }
384         if ( errors ) {
385             for ( i = 0; i <= wx+1; i++ ) {
386                 histo -> set_error( 0, i, 0 );
387                 histo -> set_error( 0, i, wy+1 );
388             }
389             for ( i = 1; i <= wy; i++ ) {
390                 histo -> set_error( 0, 0, i );
391                 histo -> set_error( 0, wx+1, i );
392             }
393         }
394     }
395     if ( errors ) {
396         for ( long x = 0; x < wx; x++ ) {
397             for ( long y = 0; y < wy; y++ ) {
398                 histo -> set_height( contents[ map_.content( x, wx, y, wy ) ],
399                                     x + off, y + off );
400                 histo -> set_error ( errors[ map_.error( x, wx, y, wy ) ],
401                                     x + off, y + off );
402             }
403         }
404     } else {
405         for ( long x = 0; x < wx; x++ ) {
406             for ( long y = 0; y < wy; y++ ) {
407                 histo -> set_height( contents[ map_.content( x, wx, y, wy ) ],
408                                     x + off, y + off );
409             }
410         }
411     }
412     OHHistogramProvider::Status s = OHHistogramProvider::publish( *histo, name );

```

```

413     delete histo;
414     return s;
415 }

```

5.16.3.3 `template<class TContent, class TError, class TAxis, class TMap>
OHRawProvider< TContent, TError, TAxis, TMap >::Status
OHRawProvider< TContent, TError, TAxis, TMap >::publish (const
string & name, const string & title, const string & xlabel, long xcount,
TAxis xoffset, TAxis xwidth, const string & ylabel, long ycount, TAxis
yoffset, TAxis ywidth, TContent * contents, TError * errors, bool
outOfRangeBins, const vector< string > & labels, const vector< string >
& values)`

Publish a 2D histogram with fixed width bins.

Parameters:

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

title The histogram title

xlabel X Axis label

xcount The number of bins along the x axis, excluding underflow and overflow

xoffset Lower edge of first bin along x axis

xwidth The (fixed) width of a bin along the x axis

ylabel Y Axis label

ycount The number of bins along the y axis, excluding underflow and overflow

yoffset Lower edge of first bin along y axis

ywidth The (fixed) width of a bin along the y axis

contents A pointer to the position where the bin contents (heights) are located.

errors A pointer to the position where the bin errors are located or 0 if no errors.

outOfRangeBins Out of range bins or not

labels Labels for any annotations that should be attached to the histogram.

values Values for any annotations that should be attached to the histogram.

Returns:

OHRawProvider::CommFailure if a communication error occurred, OHRawProvider::Invalid-Argument if one or more arguments is invalid, else OHRawProvider::Success.

See also:

OHRawProviderDM (p. 106)

Definition at line 237 of file OHRawProvider.inl.

References OHHistogramProvider::publish.

```

251                                     {
252     assert( contents );
253     OHHistogramData2D* histo = new OHHistogramData2D;
254     histo -> set_title( title );
255     histo -> set_label( xlabel, OHHistogramData2D::X );

```

```

256     histo -> set_label( ylabel, OHHistogramData2D::Y );
257     histo -> set_annotations( labels, values );
258     histo -> set_partition( xoffset, xwidth, xcount, OHHistogramData2D::X );
259     histo -> set_partition( yoffset, ywidth, ycount, OHHistogramData2D::Y );
260     long wx = xcount, wy = ycount, off;
261     if ( outOfRangeBins ) {
262         wx += 2;
263         wy += 2;
264         off = 0;
265     } else {
266         off = 1;
267         long i;
268         // Clear underflow and overflow
269         for ( i = 0; i <= wx+1; i++ ) {
270             histo -> set_height( 0, i, 0 );
271             histo -> set_height( 0, i, wy+1 );
272         }
273         for ( i = 1; i <= wy; i++ ) {
274             histo -> set_height( 0, 0, i );
275             histo -> set_height( 0, wx+1, i );
276         }
277         if ( errors ) {
278             for ( i = 0; i <= wx+1; i++ ) {
279                 histo -> set_error( 0, i, 0 );
280                 histo -> set_error( 0, i, wy+1 );
281             }
282             for ( i = 1; i <= wy; i++ ) {
283                 histo -> set_error( 0, 0, i );
284                 histo -> set_error( 0, wx+1, i );
285             }
286         }
287     }
288     if ( errors ) {
289         for ( long x = 0; x < wx; x++ ) {
290             for ( long y = 0; y < wy; y++ ) {
291                 histo -> set_height( contents[ map_.content( x, wx, y, wy ) ],
292                                     x + off, y + off );
293                 histo -> set_error ( errors[ map_.error( x, wx, y, wy ) ],
294                                    x + off, y + off );
295             }
296         }
297     } else {
298         for ( long x = 0; x < wx; x++ ) {
299             for ( long y = 0; y < wy; y++ ) {
300                 histo -> set_height( contents[ map_.content( x, wx, y, wy ) ],
301                                     x + off, y + off );
302             }
303         }
304     }
305     OHHistogramProvider::Status s = OHHistogramProvider::publish( *histo, name );
306     delete histo;
307     return s;
308 }

```

5.16.3.4 `template<class TContent, class TError, class TAxis, class TMap>`
OHRawProvider< TContent, TError, TAxis, TMap >::Status
OHRawProvider< TContent, TError, TAxis, TMap >::publish (const
string & *name*, const string & *title*, const string & *label*, long *bincount*,
TAxis * *axis*, TContent * *contents*, TError * *errors*, bool *outOfRangeBins*,
const vector< string > & *labels*, const vector< string > & *values*)

Publish a 1D histogram with variable width bins.

Parameters:

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

title The histogram title

label X Axis label

bincount The number of bins excluding underflow and overflow

axis A pointer to the axis partition (bincount + 1 values)

contents A pointer to the position where the bin contents (heights) are located.

errors A pointer to the position where the bin errors are located or 0 if no errors.

outOfRangeBins Out of range bins or not

labels Labels for any annotations that should be attached to the histogram.

values Values for any annotations that should be attached to the histogram.

Returns:

OHRawProvider::CommFailure if a communication error occurred, OHRawProvider::InvalidArgument if one or more arguments is invalid, else OHRawProvider::Success.

See also:

OHRawProviderDM (p.106)

Definition at line 157 of file OHRawProvider.inl.

References OHHistogramProvider::publish.

```

166                                     {
167     assert( contents );
168     vector<double> part;
169     OHHistogramData1D* histo = new OHHistogramData1D;
170     histo -> set_title( title );
171     histo -> set_label( label );
172     histo -> set_annotations( labels, values );
173     for ( long i = 0; i <= bincount; i++ ) part.push_back( axis[ i ] );
174     histo -> set_partition( part );
175     part.clear( );
176     long wx = bincount, off;
177     if ( outOfRangeBins ) {
178         wx += 2;
179         off = 0;
180     } else {
181         off = 1;
182         // Clear underflow and overflow
183         histo -> set_height( 0, 0 );
184         histo -> set_height( 0, wx + 1 );
185         if ( errors ) {
186             histo -> set_error( 0, 0 );
187             histo -> set_error( 0, wx + 1 );
188         }
189     }
190     if ( errors ) {
191         for ( long x = 0; x < wx; x++ ) {
192             histo -> set_height( contents[ map_.content( x, wx ) ], x + off );
193             histo -> set_error( errors[ map_.error( x, wx ) ], x + off );
194         }
195     } else {
196         for ( long x = 0; x < wx; x++ ) {
197             histo -> set_height( contents[ map_.content( x, wx ) ], x + off );
198         }

```

```

199     }
200     OHHistogramProvider::Status s = OHHistogramProvider::publish( *histo, name );
201     delete histo;
202     return s;
203 }

```

5.16.3.5 `template<class TContent, class TError, class TAxis, class TMap>
OHRawProvider< TContent, TError, TAxis, TMap >::Status
OHRawProvider< TContent, TError, TAxis, TMap >::publish (const
string & name, const string & title, const string & label, long bincount,
TAxis offset, TAxis width, TContent * contents, TError * errors, bool
outOfRangeBins, const vector< string > & labels, const vector< string >
& values)`

Publish a 1D histogram with fixed width bins.

Parameters:

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

title The histogram title

label X Axis label

bincount The number of bins excluding underflow and overflow

offset Lower edge of first bin

width The (fixed) width of a bin

contents A pointer to the position where the bin contents (heights) are located.

errors A pointer to the position where the bin errors are located or 0 if no errors.

outOfRangeBins Out of range bins or not

labels Labels for any annotations that should be attached to the histogram.

values Values for any annotations that should be attached to the histogram.

Returns:

OHRawProvider::CommFailure if a communication error occurred, OHRawProvider::InvalidArgument if one or more arguments is invalid, else OHRawProvider::Success.

See also:

OHRawProviderDM (p. 106)

Definition at line 86 of file OHRawProvider.inl.

References OHHistogramProvider::publish.

```

96                                     {
97     assert( contents );
98     OHHistogramData1D* histo = new OHHistogramData1D;
99     histo -> set_title( title );
100    histo -> set_label( label );
101    histo -> set_annotations( labels, values );
102    histo -> set_partition( offset, width, bincount );
103    long wx = bincount, off;
104    if ( outOfRangeBins ) {
105        wx += 2;
106        off = 0;

```

```

107     } else {
108         off = 1;
109         // Clear underflow and overflow
110         histo -> set_height( 0, 0 );
111         histo -> set_height( 0, wx + 1 );
112         if ( errors ) {
113             histo -> set_error( 0, 0 );
114             histo -> set_error( 0, wx + 1 );
115         }
116     }
117     if ( errors ) {
118         for ( long x = 0; x < wx; x++ ) {
119             histo -> set_height( contents[ map_.content( x, wx ) ], x + off );
120             histo -> set_error ( errors[ map_.error( x, wx ) ], x + off );
121         }
122     } else {
123         for ( long x = 0; x < wx; x++ ) {
124             histo -> set_height( contents[ map_.content( x, wx ) ], x + off );
125         }
126     }
127     OHHistogramProvider::Status s = OHHistogramProvider::publish( *histo, name );
128     delete histo;
129     return s;
130 }

```

5.16.3.6 `template<class TContent, class TError, class TAxis, class TMap> bool
OHRawProvider< TContent, TError, TAxis, TMap >::valid (void)
[virtual]`

Validity test.

Returns:

true if the provider was successfully created, otherwise false.

See also:

`operator bool()` (p. 98)

Reimplemented from `OHHistogramProvider` (p. 83).

Definition at line 56 of file `OHRawProvider.inl`.

References `OHHistogramProvider::valid`.

```

56                                     {
57     return OHHistogramProvider::valid( );
58 }

```

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

- `OHRawProvider.h`
- `OHRawProvider.inl`

5.17 OHRawProviderDM Class Reference

RAW Provider default map.

```
#include <OHRawProvider.h>
```

Public Methods

- long **content** (long x, long wx, long y=0, long wy=0, long z=0, long wz=0) const
Calculate array index for bin content.
- long **error** (long x, long wx, long y=0, long wy=0, long z=0, long wz=0) const
Calculate array index for bin error.

5.17.1 Detailed Description

RAW Provider default map.

This is the default binmap used by the **OHRawProvider** (p.97) template, it assumes that the bins are ordered according to

$$\text{index} = x + \text{width}_x * (y + \text{width}_y * z)$$

Where z=0 for 2D histograms and y=z=0 for 1D histograms. width_x and width_y is the total number (possibly including underflow and overflow) of bins in the x and y directions.

x, y and z starts at 0 and end at width_? - 1.

If underflow and overflow bins are used they are addressed with minimum and maximum values of x, y and z.

Confused? The idea here is that the OH says: I want the height (or error) corresponding to bin (x,y,z) from Your array, where is it? The functions in the map should return the correct array index corresponding to (x,y,z).

Observe the possibility to have an array of both bin contents and bin errors in a mixed order by providing the appropriate map to **OHRawProvider** (p. 97)

Example:

Lets say we have the bin contents and errors structured as

```
height1 | error1 | height2 | error2 | ...
```

in one and the same array of float values. Then we implement a new map

```
class MyMap {
public:
    long content( long x,   long wx,
                 long y=0, long wy=0,
                 long z=0, long wz=0 ) const {
        return ( x + wx * ( y + wy * z ) ) * 2;
    }
    long error( long x,   long wx,
               long y=0, long wy=0,
               long z=0, long wz=0 ) const {
        return ( x + wx * ( y + wy * z ) ) * 2 + 1;
    }
};
```

```
}
```

And then we create the provider object according to (assume float values for the axis)

```
OHRawProviderDM<float,float,float,MyMap> myprovider;  
myprovider.publish( ... );
```

In this way there is in principle no limitations to how you have the data structured.

See also:

OHRawProvider (p. 97)

Definition at line 88 of file OHRawProvider.h.

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

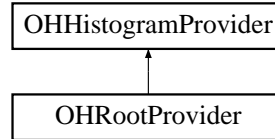
- **OHRawProvider.h**
- **OHRawProvider.inl**

5.18 OHRootProvider Class Reference

ROOT Provider.

```
#include <OHRootProvider.h>
```

Inheritance diagram for OHRootProvider::



Public Methods

- **OHRootProvider** (const IPCPartition &p, const string &server, const string &name)
Create a new ROOT provider.
- virtual **operator bool** ()
Validity test.
- virtual bool **valid** (void)
Validity test.
- Status **publish** (const TH1 &histogram, const string &name, bool update=false)
Publish a ROOT 1D histogram.
- Status **publish** (const TH2 &histogram, const string &name, bool update=false)
Publish a ROOT 2D histogram.
- Status **publish** (const TH3 &histogram, const string &name, bool update=false)
Publish a ROOT 3D histogram.

5.18.1 Detailed Description

ROOT Provider.

This class provides functionality to export histograms from ROOT to the OH

Definition at line 20 of file OHRootProvider.h.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 OHRootProvider::OHRootProvider (const IPCPartition & p, const string & server, const string & name)

Create a new ROOT provider.

Parameters:

p A valid IPCPartition

server Name of a valid OH server

name Name of the provider, user is responsible for specifying a unique name. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

Definition at line 20 of file OHRootProvider.cxx.

```

22                                     :
23                                     OHHistogramProvider( p, server, name ) {
24 }
```

5.18.3 Member Function Documentation

5.18.3.1 OHRootProvider::operator bool () [virtual]

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

`valid()` (p. 111)

Reimplemented from **OHHistogramProvider** (p. 79).

Definition at line 30 of file OHRootProvider.cxx.

References OHHistogramProvider::valid.

```

30                                     {
31     return OHHistogramProvider::valid( );
32 }
```

5.18.3.2 OHRootProvider::Status OHRootProvider::publish (const TH3 & histogram, const string & name, bool update = false)

Publish a ROOT 3D histogram.

Parameters:

histogram A valid ROOT histogram

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

update A flag which decides about histogram update (by default false)

Returns:

OHRootProvider::CommFailure if a communication error occurred, OHRootProvider::Invalid-Argument if one or more arguments is invalid, else OHRootProvider::Success.

Definition at line 85 of file OHRootProvider.cxx.

References OHHistogramProvider::publish.

```

86                                                                                                     {
87     OHHistogramData3D* h = convert( const_cast<TH3&>( histogram ) );
88     Status s = OHHistogramProvider::publish( *h, name, update );
89     delete h;
90     return s;
91 }

```

5.18.3.3 OHRootProvider::Status OHRootProvider::publish (const TH2 & histogram, const string & name, bool update = false)

Publish a ROOT 2D histogram.

Parameters:

histogram A valid ROOT histogram

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

update A flag which decides about histogram update (by default false)

Returns:

OHRootProvider::CommFailure if a communication error occurred, OHRootProvider::InvalidArgument if one or more arguments is invalid, else OHRootProvider::Success.

Definition at line 68 of file OHRootProvider.cxx.

References OHHistogramProvider::publish.

```

69                                                                                                     {
70     OHHistogramData2D* h = convert( const_cast<TH2&>( histogram ) );
71     Status s = OHHistogramProvider::publish( *h, name, update );
72     delete h;
73     return s;
74 }

```

5.18.3.4 OHRootProvider::Status OHRootProvider::publish (const TH1 & histogram, const string & name, bool update = false)

Publish a ROOT 1D histogram.

Parameters:

histogram A valid ROOT histogram

name A name describing the histogram. Should only contain characters from [a-z], [A-Z], [0-9] and '_'.

update A flag which decides about histogram update (by default false)

Returns:

OHRootProvider::CommFailure if a communication error occurred, OHRootProvider::InvalidArgument if one or more arguments is invalid, else OHRootProvider::Success.

Definition at line 51 of file OHRootProvider.cxx.

References OHHistogramProvider::publish.

```
52                                     {
53     OHHistogramData1D* h = convert( const_cast<TH1&>( histogram ) );
54     Status s = OHHistogramProvider::publish( *h, name, update );
55     delete h;
56     return s;
57 }
```

5.18.3.5 bool OHRootProvider::valid (void) [virtual]

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

operator bool() (p. 109)

Reimplemented from **OHHistogramProvider** (p. 83).

Definition at line 38 of file OHRootProvider.cxx.

References OHHistogramProvider::valid.

```
38                                     {
39     return OHHistogramProvider::valid( );
40 }
```

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

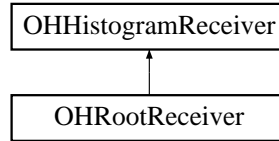
- **OHRootProvider.h**
- **OHRootProvider.cxx**

5.19 OHRootReceiver Class Reference

ROOT Receiver.

```
#include <OHRootReceiver.h>
```

Inheritance diagram for OHRootReceiver::



Protected Methods

- virtual bool **receive** (TH1 **histogram*, vector< string > &*labels*, vector< string > &*values*)

Called when a ROOT 1D histogram is received.

- virtual bool **receive** (TH2 **histogram*, vector< string > &*labels*, vector< string > &*values*)

Called when a ROOT 2D histogram is received.

- virtual bool **receive** (TH3 **histogram*, vector< string > &*labels*, vector< string > &*values*)

Called when a ROOT 3D histogram is received.

5.19.1 Detailed Description

ROOT Receiver.

This class provides functionality to import histograms from the OH to ROOT

Definition at line 25 of file OHRootReceiver.h.

5.19.2 Member Function Documentation

5.19.2.1 bool OHRootReceiver::receive (TH3 * *histogram*, vector< string > & *labels*, vector< string > & *values*) [protected, virtual]

Called when a ROOT 3D histogram is received.

Override this function in order to receive ROOT 3D histograms. User is responsible for deleting the histogram either directly or by calling the baseclass function (this function) which will delete the histogram. The histograms are created with name "default" and user is responsible for changing the name if necessary. (name != title, see ROOT reference)

Parameters:

histogram The received ROOT 3D histogram

labels Labels for any annotations associated to the histogram

values Values for any annotations associated to the histogram

Returns:

User should return true if the histogram was successfully received, otherwise false. This function returns false by default.

Definition at line 64 of file OHRootReceiver.cxx.

```

66                                     {
67     delete histogram;
68     return false; // The histogram was not received by user
69 }
```

5.19.2.2 `bool OHRootReceiver::receive (TH2 * histogram, vector< string > & labels, vector< string > & values)` [protected, virtual]

Called when a ROOT 2D histogram is received.

Override this function in order to receive ROOT 2D histograms. User is responsible for deleting the histogram either directly or by calling the baseclass function (this function) which will delete the histogram. The histograms are created with name "default" and user is responsible for changing the name if necessary. (name != title, see ROOT reference)

Parameters:

histogram The received ROOT 2D histogram

labels Labels for any annotations associated to the histogram

values Values for any annotations associated to the histogram

Returns:

User should return true if the histogram was successfully received, otherwise false. This function returns false by default.

Definition at line 45 of file OHRootReceiver.cxx.

```

47                                     {
48     delete histogram;
49     return false; // The histogram was not received by user
50 }
```

5.19.2.3 `bool OHRootReceiver::receive (TH1 * histogram, vector< string > & labels, vector< string > & values)` [protected, virtual]

Called when a ROOT 1D histogram is received.

Override this function in order to receive ROOT 1D histograms. User is responsible for deleting the histogram either directly or by calling the baseclass function (this function) which will delete the histogram. The histograms are created with name "default" and user is responsible for changing the name if necessary. (name != title, see ROOT reference)

Parameters:

histogram The received ROOT 1D histogram

labels Labels for any annotations associated to the histogram

values Values for any annotations associated to the histogram

Returns:

User should return true if the histogram was successfully received, otherwise false. This function returns false by default.

Definition at line 26 of file OHRootReceiver.cxx.

```
28                                     {
29     delete histogram;
30     return false; // The histogram was not received by user
31 }
```

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

- **OHRootReceiver.h**
- **OHRootReceiver.cxx**

5.20 OHServerIterator Class Reference

Server iterator.

```
#include <OHServerIterator.h>
```

Public Methods

- **OHServerIterator** (IPCPartition &p)

Create a new provider iterator.

- **operator bool** ()

Validity test.

- **bool valid** (void)

Validity test.

- **bool operator++** ()

Advance the iterator one position.

- **bool operator++** (int)

Advance the iterator one position.

- **bool operator--** ()

Retreat the iterator one position.

- **bool operator--** (int)

Retreat the iterator one position.

- **operator string** () const

Retrieve name of current server.

- **string name** (void) const

Retrieve name of current server.

- **void reset** (void)

Reset the iterator to its initial state.

5.20.1 Detailed Description

Server iterator.

This class provides functionality to retrieve the names of all OH servers in an IPC partition.

Definition at line 21 of file OHServerIterator.h.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 OHServerIterator::OHServerIterator (IPCPartition & p)

Create a new provider iterator.

Immediately after construction the "current server" is undefined and the iterator must be advanced with one of the ++ operators.

Parameters:

p A valid IPCPartition

Definition at line 18 of file OHServerIterator.cxx.

```

18                                     :
19                                     iterator_( p ) {
20 }
```

5.20.3 Member Function Documentation

5.20.3.1 string OHServerIterator::name (void) const

Retrieve name of current server.

Returns:

The name of the current server or "" if the current server is undefined.

Definition at line 82 of file OHServerIterator.cxx.

```

82                                     {
83     const char *n = iterator_.name( );
84     if ( n != 0 ) {
85         return n;
86     }
87     return string("");
88 }
```

5.20.3.2 OHServerIterator::operator bool ()

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

[valid\(\)](#) (p. 118)

Definition at line 26 of file OHServerIterator.cxx.

```

26                                     {
27     return true;
28 }
```

5.20.3.3 OHServerIterator::operator string () const

Retrieve name of current server.

Returns:

The name of the current server or "" if the current server is undefined.

Definition at line 70 of file OHServerIterator.cxx.

```
70                                     {
71     const char *n = iterator_.name( );
72     if ( n != 0 ) {
73         return n;
74     }
75     return string("");
76 }
```

5.20.3.4 bool OHServerIterator::operator++ (int)

Advance the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 48 of file OHServerIterator.cxx.

```
48                                     {
49     return iterator_++;
50 }
```

5.20.3.5 bool OHServerIterator::operator++ ()

Advance the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 41 of file OHServerIterator.cxx.

```
41                                     {
42     return iterator_++;
43 }
```

5.20.3.6 bool OHServerIterator::operator-- (int)

Retreat the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 62 of file OHServerIterator.cxx.

```
62                                     {
63     return iterator_--;
64 }
```

5.20.3.7 `bool OHServerIterator::operator-- ()`

Retreat the iterator one position.

Returns:

true if new position is valid, otherwise false.

Definition at line 55 of file `OHServerIterator.cxx`.

```
55                                     {  
56     return iterator_--;  
57 }
```

5.20.3.8 `bool OHServerIterator::valid (void)`

Validity test.

Returns:

true if the object was constructed successfully, otherwise false

See also:

`operator bool()` (p. 116)

Definition at line 34 of file `OHServerIterator.cxx`.

```
34                                     {  
35     return true;  
36 }
```

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

- `OHServerIterator.h`
- `OHServerIterator.cxx`

Chapter 6

OH Core Page Documentation

6.1 Naming conventions

The unique identifier used to identify a histogram or set of histograms currently has the following structure:

```
servername.OH:providername:histogramname:YYYYMMDDhhmmss:counter
```

Where `servername` is the name of an OH (IS) server, `providername` is the name of the provider who published the histogram, `histogramname` is a name describing the histogram and `YYYYMMDDhhmmss` is the time when the histogram was published. Counter is a 'collision counter' used to make the identifier unique, the value of this counter does NOT imply a specific ordering among histograms with otherwise identical identifiers.

The histogram providers publish their name in the OH (IS) server which they belong to upon construction, the name is published in an `ISInfoString` information type with the following name:

```
servername.OHP:providername
```

Where `servername` is the name of an IS server, `providername` is the name of the provider who is activated. The information type is removed from the server when the provider is destroyed.

6.2 The ROOT Histogram Provider

The ROOT histogram provider is capable of exporting ROOT TH1, TH2 and TH3 histogram objects or any object which has one of these as an ancestor.

The provider will export the following information to the OH

- histogram title
- bin heights for all bins, including underflow and overflow
- bin errors for all bins, including underflow and overflow
- axis labels
- axis partitions
- rms statistics value for each axis
- mean statistics value for each axis

No annotations are associated to ROOT histograms.

For more information on how to export histograms from ROOT see **OHRootProvider** (p.108).

6.3 The ROOT Histogram Receiver

The ROOT histogram receiver is capable of importing histograms from the OH to ROOT TH1, TH2 and TH3 histogram objects.

The receiver will import the following information

- histogram title
- bin heights for all bins, including underflow and overflow
- bin errors for all bins, including underflow and overflow
- axis labels
- axis partitions

If the imported histogram was tagged with one or more annotations they will be received as separate arguments.

The RMS and mean statistics values cannot be set directly in a ROOT histogram because these are computed from various other parameters. The RMS and mean values displayed in the statistics box in an imported histogram is thus not the values received from the OH but approximate values calculated from the bin heights by ROOT.

Todo:

Allow user to receive the original RMS and mean values.

For more information on how to import histograms to ROOT see **OHRootReceiver** (p. 112).

6.4 Todo List

Class ISHistogramSet1D Find a way to serialize sets with IS (ISInfoAny? ISInfo Arrays?)

Class ISHistogramSet2D Find a way to serialize sets with IS (ISInfoAny? ISInfo Arrays?)

Class ISHistogramSet3D Find a way to serialize sets with IS (ISInfoAny? ISInfo Arrays?)

Class OHRawProvider Support 3D histograms if necessary.

Page The ROOT Histogram Receiver Allow user to receive the original RMS and mean values.

Index

- ~ISHistogram1D
 - ISHistogram1D, 9
 - ~ISHistogram2D
 - ISHistogram2D, 14
 - ~ISHistogram3D
 - ISHistogram3D, 19
 - ~ISHistogramSet1D
 - ISHistogramSet1D, 25
 - ~ISHistogramSet2D
 - ISHistogramSet2D, 27
 - ~ISHistogramSet3D
 - ISHistogramSet3D, 29
 - ~OHHistogramData
 - OHHistogramData, 31
 - ~OHHistogramIterator
 - OHHistogramIterator, 70
 - ~OHHistogramProvider
 - OHHistogramProvider, 77
 - ~OHHistogramReceiver
 - OHHistogramReceiver, 85
 - ANY_DAY
 - OHHistogramIterator, 71
 - OHHistogramSubscriber, 88
 - ANY_HOUR
 - OHHistogramIterator, 71
 - OHHistogramSubscriber, 88
 - ANY_MINUTE
 - OHHistogramIterator, 71
 - OHHistogramSubscriber, 88
 - ANY_MONTH
 - OHHistogramIterator, 71
 - OHHistogramSubscriber, 88
 - ANY_SECOND
 - OHHistogramIterator, 71
 - OHHistogramSubscriber, 88
 - ANY_YEAR
 - OHHistogramIterator, 71
 - OHHistogramSubscriber, 88
 - axis_type
 - OHHistogramData, 33
 - OHHistogramData1D, 40
 - OHHistogramData2D, 50
 - OHHistogramData3D, 61
 - bin_count
 - OHHistogramData, 33
 - OHHistogramData1D, 40
 - OHHistogramData2D, 50
 - OHHistogramData3D, 61
 - content
 - OHRawProviderDM, 106
 - error
 - OHRawProviderDM, 106
 - error_type
 - OHHistogramData, 32
 - OHHistogramData1D, 39
 - OHHistogramData2D, 49
 - OHHistogramData3D, 60
 - get_annotations
 - OHHistogramData, 33
 - OHHistogramData1D, 41
 - OHHistogramData2D, 51
 - OHHistogramData3D, 62
 - get_error
 - OHHistogramData, 34
 - OHHistogramData1D, 41
 - OHHistogramData2D, 51
 - OHHistogramData3D, 62
 - get_height
 - OHHistogramData, 34
 - OHHistogramData1D, 42
 - OHHistogramData2D, 52
 - OHHistogramData3D, 63
 - get_label
 - OHHistogramData, 31
 - OHHistogramData1D, 38
 - OHHistogramData2D, 48
 - OHHistogramData3D, 59
 - get_mean
 - OHHistogramData, 32
 - OHHistogramData1D, 39
 - OHHistogramData2D, 49
 - OHHistogramData3D, 60
 - get_partition
 - OHHistogramData, 34
 - OHHistogramData1D, 42
-

- OHHistogramData2D, 52, 53
- OHHistogramData3D, 63, 64
- get_pmerror
 - OHHistogramData, 35
 - OHHistogramData1D, 43
 - OHHistogramData2D, 54
 - OHHistogramData3D, 65
- get_rms
 - OHHistogramData, 33
 - OHHistogramData1D, 39
 - OHHistogramData2D, 49
 - OHHistogramData3D, 60
- get_title
 - OHHistogramData, 31
 - OHHistogramData1D, 38
 - OHHistogramData2D, 48
 - OHHistogramData3D, 59
- is_annotatations
 - ISHistogram1D, 10
 - ISHistogram2D, 15
 - ISHistogram3D, 20
- is_annotatations_size
 - ISHistogram1D, 10
 - ISHistogram2D, 15
 - ISHistogram3D, 20
- is_bins
 - ISHistogram1D, 10
 - ISHistogram2D, 15
 - ISHistogram3D, 20
- is_bins_size
 - ISHistogram1D, 10
 - ISHistogram2D, 15
 - ISHistogram3D, 20
- is_errors
 - ISHistogram1D, 11
 - ISHistogram2D, 15
 - ISHistogram3D, 21
- is_errors_size
 - ISHistogram1D, 11
 - ISHistogram2D, 16
 - ISHistogram3D, 21
- is_minusererrors
 - ISHistogram1D, 11
 - ISHistogram2D, 16
 - ISHistogram3D, 21
- is_minusererrors_size
 - ISHistogram1D, 11
 - ISHistogram2D, 16
 - ISHistogram3D, 21
- is_plusererrors
 - ISHistogram1D, 11
 - ISHistogram2D, 16
 - ISHistogram3D, 21
- is_plusererrors_size
 - ISHistogram1D, 11
 - ISHistogram2D, 16
 - ISHistogram3D, 21
- is_title
 - ISHistogram1D, 12
 - ISHistogram2D, 16
 - ISHistogram3D, 22
- is_xaxis
 - ISHistogram1D, 12
 - ISHistogram2D, 17
 - ISHistogram3D, 22
- is_xaxis_size
 - ISHistogram1D, 12
 - ISHistogram2D, 17
 - ISHistogram3D, 22
- is_xlabel
 - ISHistogram1D, 12
 - ISHistogram2D, 17
 - ISHistogram3D, 22
- is_xmean
 - ISHistogram1D, 12
 - ISHistogram2D, 17
 - ISHistogram3D, 22
- is_xrms
 - ISHistogram1D, 12
 - ISHistogram2D, 17
 - ISHistogram3D, 22
- is_xtype
 - ISHistogram1D, 13
 - ISHistogram2D, 17
 - ISHistogram3D, 23
- is_yaxis
 - ISHistogram2D, 18
 - ISHistogram3D, 23
- is_yaxis_size
 - ISHistogram2D, 18
 - ISHistogram3D, 23
- is_ylabel
 - ISHistogram2D, 18
 - ISHistogram3D, 23
- is_ymean
 - ISHistogram2D, 18
 - ISHistogram3D, 23
- is_yrms
 - ISHistogram2D, 18
 - ISHistogram3D, 23
- is_ytype
 - ISHistogram2D, 18
 - ISHistogram3D, 23
- is_zaxis
 - ISHistogram3D, 24
- is_zaxis_size
 - ISHistogram3D, 24

- is_zlabel
 - ISHistogram3D, 24
- is_zmean
 - ISHistogram3D, 24
- is_zrms
 - ISHistogram3D, 24
- is_ztype
 - ISHistogram3D, 24
- ISHistogram1D, 9
 - ~ISHistogram1D, 9
 - is_annotations, 10
 - is_annotations_size, 10
 - is_bins, 10
 - is_bins_size, 10
 - is_errors, 11
 - is_errors_size, 11
 - is_minusererrors, 11
 - is_minusererrors_size, 11
 - is_plusererrors, 11
 - is_plusererrors_size, 11
 - is_title, 12
 - is_xaxis, 12
 - is_xaxis_size, 12
 - is_xlabel, 12
 - is_xmean, 12
 - is_xrms, 12
 - is_xtype, 13
 - ISHistogram1D, 9, 10
 - publishGuts, 10
 - refreshGuts, 10
- ISHistogram2D, 14
 - ~ISHistogram2D, 14
 - is_annotations, 15
 - is_annotations_size, 15
 - is_bins, 15
 - is_bins_size, 15
 - is_errors, 15
 - is_errors_size, 16
 - is_minusererrors, 16
 - is_minusererrors_size, 16
 - is_plusererrors, 16
 - is_plusererrors_size, 16
 - is_title, 16
 - is_xaxis, 17
 - is_xaxis_size, 17
 - is_xlabel, 17
 - is_xmean, 17
 - is_xrms, 17
 - is_xtype, 17
 - is_yaxis, 18
 - is_yaxis_size, 18
 - is_ylabel, 18
 - is_ymean, 18
 - is_yrms, 18
 - is_ytype, 18
 - ISHistogram2D, 14, 15
 - publishGuts, 15
 - refreshGuts, 15
- ISHistogram3D, 19
 - ~ISHistogram3D, 19
 - is_annotations, 20
 - is_annotations_size, 20
 - is_bins, 20
 - is_bins_size, 20
 - is_errors, 21
 - is_errors_size, 21
 - is_minusererrors, 21
 - is_minusererrors_size, 21
 - is_plusererrors, 21
 - is_plusererrors_size, 21
 - is_title, 22
 - is_xaxis, 22
 - is_xaxis_size, 22
 - is_xlabel, 22
 - is_xmean, 22
 - is_xrms, 22
 - is_xtype, 23
 - is_yaxis, 23
 - is_yaxis_size, 23
 - is_ylabel, 23
 - is_ymean, 23
 - is_yrms, 23
 - is_ytype, 23
 - is_zaxis, 24
 - is_zaxis_size, 24
 - is_zlabel, 24
 - is_zmean, 24
 - is_zrms, 24
 - is_ztype, 24
 - ISHistogram3D, 19, 20
 - publishGuts, 20
 - refreshGuts, 20
- ISHistogramSet1D
 - ~ISHistogramSet1D, 25
 - ISHistogramSet1D, 25
 - publishGuts, 25
 - refreshGuts, 25
- ISHistogramSet1D, 25
 - ISHistogramSet1D, 25
- ISHistogramSet2D
 - ~ISHistogramSet2D, 27
 - ISHistogramSet2D, 27
 - publishGuts, 27
 - refreshGuts, 27
- ISHistogramSet2D, 27
 - ISHistogramSet2D, 27
- ISHistogramSet3D
 - ~ISHistogramSet3D, 29

- ISHistogramSet3D, 29
 - publishGuts, 29
 - refreshGuts, 29
- ISHistogramSet3D, 29
 - ISHistogramSet3D, 29
- map-
 - OHRawProvider, 97
- name
 - OHHistogramIterator, 73
 - OHPProviderIterator, 94
 - OHServerIterator, 116
- oh_subscriber_is_callback
 - OHHistogramSubscriber, 88
- OHHistogramData
 - ~OHHistogramData, 31
 - error_type, 32
 - get_label, 31
 - get_mean, 32
 - get_rms, 33
 - get_title, 31
 - set_label, 31
 - set_mean, 32
 - set_rms, 33
 - set_title, 31
- OHHistogramData, 31
 - axis_type, 33
 - bin_count, 33
 - get_annotations, 33
 - get_error, 34
 - get_height, 34
 - get_partition, 34
 - get_pmerror, 35
 - Overflow, 37
 - reset_bins, 35
 - set_annotations, 35
 - set_error, 35
 - set_height, 36
 - set_partition, 36
 - set_pmerror, 36
 - Underflow, 37
- OHHistogramData1D
 - error_type, 39
 - get_label, 38
 - get_mean, 39
 - get_rms, 39
 - get_title, 38
 - OHHistogramData1D, 38
 - set_label, 38
 - set_mean, 39
 - set_rms, 39
 - set_title, 38
- OHHistogramData1D, 38
 - axis_type, 40
 - bin_count, 40
 - get_annotations, 41
 - get_error, 41
 - get_height, 42
 - get_partition, 42
 - get_pmerror, 43
 - reset_bins, 44
 - set_annotations, 44
 - set_error, 44
 - set_height, 45
 - set_partition, 45, 46
 - set_pmerror, 46
- OHHistogramData2D
 - error_type, 49
 - get_label, 48
 - get_mean, 49
 - get_rms, 49
 - get_title, 48
 - OHHistogramData2D, 48
 - set_label, 48
 - set_mean, 49
 - set_rms, 49
 - set_title, 48
- OHHistogramData2D, 48
 - axis_type, 50
 - bin_count, 50
 - get_annotations, 51
 - get_error, 51
 - get_height, 52
 - get_partition, 52, 53
 - get_pmerror, 54
 - reset_bins, 54
 - set_annotations, 55
 - set_error, 55
 - set_height, 56
 - set_partition, 56, 57
 - set_pmerror, 57
- OHHistogramData3D
 - error_type, 60
 - get_label, 59
 - get_mean, 60
 - get_rms, 60
 - get_title, 59
 - OHHistogramData3D, 59
 - set_label, 59
 - set_mean, 60
 - set_rms, 60
 - set_title, 59
- OHHistogramData3D, 59
 - axis_type, 61
 - bin_count, 61
 - get_annotations, 62

- get_error, 62
- get_height, 63
- get_partition, 63, 64
- get_pmerror, 65
- reset_bins, 65
- set_annotations, 66
- set_error, 66
- set_height, 67
- set_partition, 67, 68
- set_pmerror, 69
- OHHistogramIterator
 - ~OHHistogramIterator, 70
 - ANY_DAY, 71
 - ANY_HOUR, 71
 - ANY_MINUTE, 71
 - ANY_MONTH, 71
 - ANY_SECOND, 71
 - ANY_YEAR, 71
 - OHHistogramIterator, 71
 - OHHistogramReceiver, 85
- OHHistogramIterator, 70
 - name, 73
 - OHHistogramIterator, 71
 - operator bool, 73
 - operator++, 73, 74
 - operator-, 74
 - provider, 74
 - retrieve, 75
 - time, 75
 - valid, 76
- OHHistogramProvider
 - ~OHHistogramProvider, 77
 - OHHistogramProvider, 78
- OHHistogramProvider, 77
 - OHHistogramProvider, 78
 - operator bool, 79
 - publish, 79, 81, 82
 - valid, 83
- OHHistogramReceiver
 - ~OHHistogramReceiver, 85
 - OHHistogramIterator, 85
 - OHHistogramSubscriber, 85
- OHHistogramReceiver, 85
 - operator bool, 86
 - receive_, 86, 87
 - valid, 87
- OHHistogramSubscriber
 - ANY_DAY, 88
 - ANY_HOUR, 88
 - ANY_MINUTE, 88
 - ANY_MONTH, 88
 - ANY_SECOND, 88
 - ANY_YEAR, 88
 - oh_subscriber_is_callback, 88
 - OHHistogramReceiver, 85
 - OHHistogramSubscriber, 89
 - operator bool, 91
 - start, 91
 - stop, 91
 - valid, 91
- OHPProviderIterator
 - OHPProviderIterator, 94
 - reset, 93
- OHPProviderIterator, 93
 - name, 94
 - OHPProviderIterator, 94
 - operator bool, 94
 - operator string, 95
 - operator++, 95
 - operator-, 95, 96
 - valid, 96
- OHRawProvider
 - map_, 97
 - OHRawProvider, 98
- OHRawProvider, 97
 - OHRawProvider, 98
 - operator bool, 98
 - publish, 99, 101, 102, 104
 - valid, 105
- OHRawProviderDM
 - content, 106
 - error, 106
- OHRawProviderDM, 106
- OHRootProvider
 - OHRootProvider, 108
- OHRootProvider, 108
 - OHRootProvider, 108
 - operator bool, 109
 - publish, 109, 110
 - valid, 111
- OHRootReceiver, 112
 - receive, 112, 113
- OHServerIterator
 - OHServerIterator, 116
 - reset, 115
- OHServerIterator, 115
 - name, 116
 - OHServerIterator, 116
 - operator bool, 116
 - operator string, 116
 - operator++, 117
 - operator-, 117
 - valid, 118
- operator bool
 - OHHistogramIterator, 73
 - OHHistogramProvider, 79

- OHHistogramReceiver, 86
- OHHistogramSubscriber, 91
- OHPProviderIterator, 94
- OHRawProvider, 98
- OHRootProvider, 109
- OHServerIterator, 116
- operator string
 - OHPProviderIterator, 95
 - OHServerIterator, 116
- operator++
 - OHHistogramIterator, 73, 74
 - OHPProviderIterator, 95
 - OHServerIterator, 117
- operator-
 - OHHistogramIterator, 74
 - OHPProviderIterator, 95, 96
 - OHServerIterator, 117
- Overflow
 - OHHistogramData, 37
- provider
 - OHHistogramIterator, 74
- publish
 - OHHistogramProvider, 79, 81, 82
 - OHRawProvider, 99, 101, 102, 104
 - OHRootProvider, 109, 110
- publishGuts
 - ISHistogram1D, 10
 - ISHistogram2D, 15
 - ISHistogram3D, 20
 - ISHistogramSet1D, 25
 - ISHistogramSet2D, 27
 - ISHistogramSet3D, 29
- receive
 - OHRootReceiver, 112, 113
- receive_
 - OHHistogramReceiver, 86, 87
- refreshGuts
 - ISHistogram1D, 10
 - ISHistogram2D, 15
 - ISHistogram3D, 20
 - ISHistogramSet1D, 25
 - ISHistogramSet2D, 27
 - ISHistogramSet3D, 29
- reset
 - OHPProviderIterator, 93
 - OHServerIterator, 115
- reset_bins
 - OHHistogramData, 35
 - OHHistogramData1D, 44
 - OHHistogramData2D, 54
 - OHHistogramData3D, 65
- retrieve
 - OHHistogramIterator, 75
- set_annotations
 - OHHistogramData, 35
 - OHHistogramData1D, 44
 - OHHistogramData2D, 55
 - OHHistogramData3D, 66
- set_error
 - OHHistogramData, 35
 - OHHistogramData1D, 44
 - OHHistogramData2D, 55
 - OHHistogramData3D, 66
- set_height
 - OHHistogramData, 36
 - OHHistogramData1D, 45
 - OHHistogramData2D, 56
 - OHHistogramData3D, 67
- set_label
 - OHHistogramData, 31
 - OHHistogramData1D, 38
 - OHHistogramData2D, 48
 - OHHistogramData3D, 59
- set_mean
 - OHHistogramData, 32
 - OHHistogramData1D, 39
 - OHHistogramData2D, 49
 - OHHistogramData3D, 60
- set_partition
 - OHHistogramData, 36
 - OHHistogramData1D, 45, 46
 - OHHistogramData2D, 56, 57
 - OHHistogramData3D, 67, 68
- set_perror
 - OHHistogramData, 36
 - OHHistogramData1D, 46
 - OHHistogramData2D, 57
 - OHHistogramData3D, 69
- set_rms
 - OHHistogramData, 33
 - OHHistogramData1D, 39
 - OHHistogramData2D, 49
 - OHHistogramData3D, 60
- set_title
 - OHHistogramData, 31
 - OHHistogramData1D, 38
 - OHHistogramData2D, 48
 - OHHistogramData3D, 59
- start
 - OHHistogramSubscriber, 91
- stop
 - OHHistogramSubscriber, 91
- time
 - OHHistogramIterator, 75

Underflow

- OHHistogramData, 37

valid

- OHHistogramIterator, 76
- OHHistogramProvider, 83
- OHHistogramReceiver, 87
- OHHistogramSubscriber, 91
- OHProviderIterator, 96
- OHRawProvider, 105
- OHRootProvider, 111
- OHServerIterator, 118