fplot

1.2.0

Generated by Doxygen 1.8.11

Contents

1 Main Page

1.1 Introduction

FPLOT is a Fortran library providing a means of interacting with `Gnuplot` from a Fortran program. The library is designed in an object-oriented manner, and as such utilizes language features that require a compiler that supports the 2003 and 2008 standards. Additionally, it is expected that Gnuplot is installed on the system path. For full functionallity, a minimum of Gnuplot v5.2 is expected.

2 Modules Index

2.1 Modules List

Here is a list of all documented modules with brief descriptions:

3 Data Type Index

3.1 Class Hierarchy

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

4 Data Type Index

4.1 Data Types List

Here are the data types with brief descriptions:

5 Module Documentation

5.1 fplot_core Module Reference

fplot_core

Data Types

- interface cm_get_string_result

  Retrieves a string from a colormap.
- type color

  Constructs a linearly spaced array.
- type colormap

  A colormap object for a surface plot.
- type cool_colormap

  Defines a colormap consisting of "cool" colors.
- interface get_string_result

  Retrieves a string from a plot_object.
- type hot_colormap

  Defines a colormap consisting of "hot" colors.
- type latex_terminal

  Defines a GNUPLOT LATEX terminal object.
- type legend

  Defines a legend object.
- interface pa_get_string_result

  Retrieves a string from a plot_axis.
- interface pd_get_string_result

  Retrieves a string from a plot_data object.

Variables

- integer(int32), parameter, public gnuplot_terminal_win32 = 1

  Defines a Win32 terminal.
- integer(int32), parameter, public gnuplot_terminal_wxt = 2

  Defines a WXT terminal.
- integer(int32), parameter, public gnuplot_terminal_qt = 3

  Defines a QT terminal.
- integer(int32), parameter, public gnuplot_terminal_png = 4

  Defines a PNG terminal.
- integer(int32), parameter, public gnuplot_terminal_latex = 5

  Defines a LATEX terminal.
- integer(int32), parameter, public marker_plus = 1

  Defines a + data point marker.
- integer(int32), parameter, public marker_x = 2

  Defines an x data point marker.
- integer(int32), parameter, public marker_asterisk = 3

  Defines an $*$ data point marker.
- integer(int32), parameter, public marker_empty_square = 4

  Defines an empty square-shaped data point marker.
- integer(int32), parameter, public marker_filled_square = 5

  Defines an filled square-shaped data point marker.
- integer(int32), parameter, public marker_empty_circle = 6

  Defines an empty circle-shaped data point marker.
- integer(int32), parameter, public marker_filled_circle = 7

  Defines an filled circle-shaped data point marker.
- integer(int32), parameter, public marker_empty_triangle = 8

  Defines an empty triangle-shaped data point marker.
- integer(int32), parameter, public marker_filled_triangle = 9

  Defines an filled triangle-shaped data point marker.
- integer(int32), parameter, public marker_empty_nabla = 10

  Defines an empty nabla-shaped data point marker.
- integer(int32), parameter, public marker_filled_nabla = 11

  Defines an filled nabla-shaped data point marker.
- integer(int32), parameter, public marker_empty_rhombus = 12

  Defines an empty rhombus-shaped data point marker.
- integer(int32), parameter, public marker_filled_rhombus = 13

  Defines an filled rhombus-shaped data point marker.
- integer(int32), parameter, public line_solid = 1

  Defines a solid line.
- integer(int32), parameter, public line_dashed = 2

  Defines a dashed line.
- integer(int32), parameter, public line_dotted = 3

  Defines a dotted line.
- integer(int32), parameter, public line_dash_dotted = 4

  Defines a dash-dotted line.
- integer(int32), parameter, public line_dash_dot_dot = 5

  Defines a dash-dot-dotted line.
- character(len=$*$), parameter, public legend_top = "top"

  Defines the legend should be placed at the top of the plot.
- character(len=$*$), parameter, public legend_center = "center"

Defines the legend should be centered on the plot.
- character(len=∗), parameter, public legend_left = "left"

  Defines the legend should be placed at the left of the plot.
- character(len=∗), parameter, public legend_right = "right"

  Defines the legend should be placed at the right of the plot.
- character(len=∗), parameter, public legend_bottom = "bottom"

  Defines the legend should be placed at the bottom of the plot.
- integer(int32), parameter, public plotdata_max_name_length = 128

  Defines the maximum number of characters allowed in a graph label.
- integer(int32), parameter gnuplot_default_window_width = 640

  The default GNUPLOT window width, in pixels.
- integer(int32), parameter gnuplot_default_window_height = 420

  The default GNUPLOT window height, in pixels.
- integer(int32), parameter gnuplot_max_label_length = 128

  Defines the maximum number of characters allowed in a graph label.
- character(len=∗), parameter gnuplot_default_fontname = "Calibri"

  Defines the default font used by text on the graph.
- integer(int32), parameter gnuplot_default_font_size = 10

  Defines the default font size used by text on the graph.
- integer(int32), parameter gnuplot_max_path_length = 256

  Defines the maximum number of characters allowed in a file path.
- type(color), parameter, public clr_black = color(0, 0, 0)

  Defines a black color.
- type(color), parameter, public clr_white = color(255, 255, 255)

  Defines a white color.
- type(color), parameter, public clr_red = color(255, 0, 0)

  Defines a red color.
- type(color), parameter, public clr_lime = color(0, 255, 0)

  Defines a lime color.
- type(color), parameter, public clr_blue = color(0, 0, 255)

  Defines a blue color.
- type(color), parameter, public clr_yellow = color(255, 255, 0)

  Defines a yellow color.
- type(color), parameter, public clr_cyan = color(0, 255, 255)

  Defines a cyan color.
- type(color), parameter, public clr_magenta = color(255, 0, 255)

  Defines a magenta color.
- type(color), parameter, public clr_silver = color(192, 192, 192)

  Defines a silver color.
- type(color), parameter, public clr_gray = color(128, 128, 128)

  Defines a gray color.
- type(color), parameter, public clr_maroon = color(128, 0, 0)

  Defines a maroon color.
- type(color), parameter, public clr_olive = color(128, 128, 0)

  Defines a olive color.
- type(color), parameter, public clr_green = color(0, 128, 0)

  Defines a green color.
- type(color), parameter, public clr_purple = color(128, 0, 128)

  Defines a purple color.
- type(color), parameter, public clr_teal = color(0, 128, 128)

  Defines a teal color.
- type(color), parameter, public clr_navy = color(0, 0, 128)

  Defines a navy color.

5.1.1 Detailed Description

fplot_core

Purpose

Provides types and routines specific necessary to support plotting operations.

5.2 fplot_errors Module Reference

plot_errors

Variables

- integer(int32), parameter plot_out_of_memory_error = 1000

  Occurs if there is insufficient memory available for the requested operation.
- integer(int32), parameter plot_invalid_input_error = 1001

  Occurs if an invalid input is provided.
- integer(int32), parameter plot_invalid_operation_error = 1002

  Occurs if an attempt is made to perform an invalid operation.
- integer(int32), parameter plot_array_size_mismatch_error = 1003

  Occurs if there is an array size mismatch error.
- integer(int32), parameter plot_gnuplot_file_error = 1004

  Occurs if there is a GNUPLOT file error.

5.2.1 Detailed Description

plot_errors

Purpose

Provides error codes for plot routines.

6 Data Type Documentation

6.1 fplot_core::cm_get_string_result Interface Reference

Retrieves a string from a colormap.

Private Member Functions

- character(len=:) function, allocatable **cm_get_string_result** (this)

6.1.1 Detailed Description

Retrieves a string from a colormap.

Parameters

in	*this*	The colormap object.

Returns

The string.

Definition at line 6774 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.2 fplot_core::color Type Reference

Constructs a linearly spaced array.

Public Member Functions

- procedure, pass, public to_hex_string => clr_to_hex_string

  Returns the color in hexadecimal format.
- procedure, pass, public copy_from => clr_copy_from

  Copies another color to this color.

Public Attributes

- integer(int32), public red = 0

  The red component of the color (must be between 0 and 255).
- integer(int32), public green = 0

  The green component of the color (must be between 0 and 255).
- integer(int32), public blue = 255

  The blue component of the color (must be between 0 and 255).

6.2.1 Detailed Description

Constructs a linearly spaced array.

Parameters

in	*start*	The first value in the array.
in	*finish*	The last value in the array.
in	*npts*	The number of values in the array.

Returns

The resulting array. Constructs two matrices (X and Y) from x and y data arrays.

Parameters

`in`	*x*	An M-element array of x data points.
`in`	*y*	An N-element array of y data points.

Returns

An N-by-M-by-2 array containing the x data matrix on the first page of the array, and the y data matrix on the second page. Describes an RGB color.

Definition at line 236 of file fplot_core.f90.

6.2.2 Member Function/Subroutine Documentation

6.2.2.1 procedure, pass, public fplot_core::color::copy_from ( )

Copies another color to this color.

Syntax

```
subroutine copy_from(class(color) this, class(color) clr)
```

Parameters

`in,out`	*this*	The color object.
`in`	*clr*	The color to copy.

Example

```
program example
    use fplot_core
    implicit none

    type(color) :: clr1, clr2

    ! Copy clr1 to clr2
    call clr2%copy_from(clr1)
end program
```

Definition at line 290 of file fplot_core.f90.

6.2.2.2 procedure, pass, public fplot_core::color::to_hex_string ( )

Returns the color in hexadecimal format.

Syntax

```
pure character(6) function clr_to_hex_string(class(color) this)
```

Parameters

in	*this*	The color object.

Returns

A string containing the hexadecimal equivalent.

Example

```
program example
    use fplot_core
    implicit none

    type(color) :: clr
    character(6) :: hex_str

    ! Return the hexadecimal form of the color
    hex_str = clr%to_hex_string()
end program
```

Definition at line 267 of file fplot_core.f90.

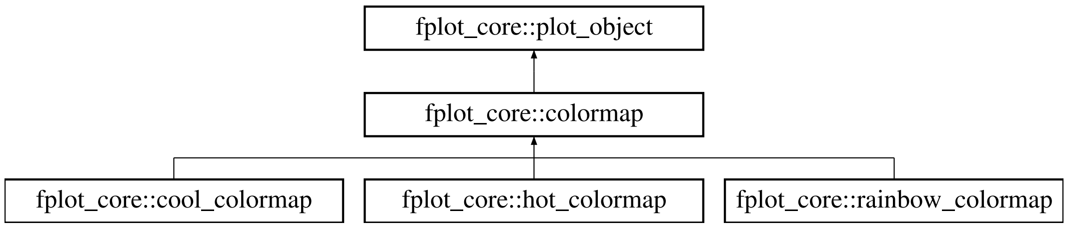
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.3 fplot_core::colormap Type Reference

A colormap object for a surface plot.

Inheritance diagram for fplot_core::colormap:



Public Member Functions

- procedure, public get_command_string => cm_get_cmd

  Gets the GNUPLOT command string to represent this colormap object.
- procedure(cm_get_string_result), deferred, public get_color_string

  Gets the GNUPLOT string defining the color distribution. For instance, this routine could return the string: '0 "dark-blue", 1 "blue", 2 "cyan", 3 "green", 4 "yellow", 5 "orange", 6 "red", 7 "dark-red"'. This string would result in a rainbow type map.

6.3.1 Detailed Description

A colormap object for a surface plot.

Definition at line 2805 of file fplot_core.f90.

6.3.2 Member Function/Subroutine Documentation

6.3.2.1 procedure, public fplot_core::colormap::get_command_string ( )

Gets the GNUPLOT command string to represent this colormap object.

Syntax

```
character(len = :) function, allocatable :: get_command_string(class(colormap) this)
```

Parameters

in	*this*	The colormap object.

Returns

The command string.

Definition at line 2817 of file fplot_core.f90.

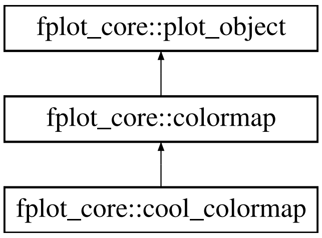
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.4 fplot_core::cool_colormap Type Reference

Defines a colormap consisting of "cool" colors.

Inheritance diagram for fplot_core::cool_colormap:



Public Member Functions

- procedure, public get_color_string => ccm_get_clr

  Gets the GNUPLOT string defining the color distribution.

6.4.1 Detailed Description

Defines a colormap consisting of "cool" colors.

Example

The following example illustrates a surface plot using a rainbow colormap.

```fortran
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    type(cool_colormap) :: map ! Using a cool colormap
    class(plot_axis), pointer :: xaxis, yaxis, zaxis

    ! Define the data
    xdata = linspace(xmin, xmax, n)
    ydata = linspace(ymin, ymax, m)
    xy = meshgrid(xdata, ydata)
    x => xy(:,:,1)
    y => xy(:,:,2)

    ! Define the function to plot
    z = sin(sqrt(x**2 + y**2))

    ! Create the plot
    call plt%initialize()
    call plt%set_colormap(map)

    ! Define titles
    call plt%set_title("Surface Example Plot 1")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    zaxis => plt%get_z_axis()
    call zaxis%set_title("Z Axis")

    ! Define the data set
    call d1%define_data(x, y, z)
    call d1%set_name("sin(sqrt(x**2 + y**2))")
    call plt%push(d1)

    ! Let GNUPLOT draw the plot
    call plt%draw()
end program
```

Definition at line 3054 of file fplot_core.f90.

6.4.2 Member Function/Subroutine Documentation

6.4.2.1 procedure, public fplot_core::cool_colormap::get_color_string ( )

Gets the GNUPLOT string defining the color distribution.

Syntax

```fortran
character(len = :) function, allocatable get_color_string(class(cool_colormap) this)
```

Parameters

in	*this*	The cool_colormap object.

Returns

The command string.

Definition at line 3065 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.5 fplot_core::get_string_result Interface Reference

Retrieves a string from a plot_object.

Private Member Functions

- character(len=:) function, allocatable **get_string_result** (this)

6.5.1 Detailed Description

Retrieves a string from a plot_object.

Parameters

in	*this*	The plot_object object.

Returns

The string.

Definition at line 6687 of file fplot_core.f90.

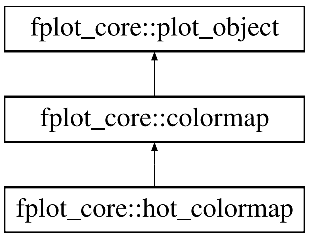
The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.6 fplot_core::hot_colormap Type Reference

Defines a colormap consisting of "hot" colors.

Inheritance diagram for fplot_core::hot_colormap:



Public Member Functions

- procedure, public get_color_string => hcm_get_clr

  Gets the GNUPLOT string defining the color distribution.

6.6.1 Detailed Description

Defines a colormap consisting of "hot" colors.

Example

The following example illustrates a surface plot using a rainbow colormap.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    type(hot_colormap) :: map ! Using a hot colormap
    class(plot_axis), pointer :: xaxis, yaxis, zaxis

    ! Define the data
    xdata = linspace(xmin, xmax, n)
    ydata = linspace(ymin, ymax, m)
    xy = meshgrid(xdata, ydata)
    x => xy(:,:,1)
    y => xy(:,:,2)

    ! Define the function to plot
    z = sin(sqrt(x**2 + y**2))

    ! Create the plot
    call plt%initialize()
    call plt%set_colormap(map)

    ! Define titles
    call plt%set_title("Surface Example Plot 1")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    zaxis => plt%get_z_axis()
    call zaxis%set_title("Z Axis")
```

```
        ! Define the data set
        call d1%define_data(x, y, z)
        call d1%set_name("sin(sqrt(x**2 + y**2))")
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 2973 of file fplot_core.f90.

6.6.2 Member Function/Subroutine Documentation

6.6.2.1 procedure, public fplot_core::hot_colormap::get_color_string ( )

Gets the GNUPLOT string defining the color distribution.

Syntax

```
    character(len = :) function, allocatable get_color_string(class(hot_colormap) this)
```

Parameters

in	*this*	The hot_colormap object.

Returns

The command string.

Definition at line 2984 of file fplot_core.f90.

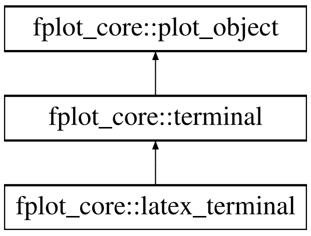
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.7 fplot_core::latex_terminal Type Reference

Defines a GNUPLOT LATEX terminal object.

Inheritance diagram for fplot_core::latex_terminal:



Public Member Functions

- procedure, public get_filename => tex_get_filename

    Gets the filename for the output LATEX file.
- procedure, public set_filename => tex_set_filename

    Sets the filename for the output LATEX file.
- procedure, public get_id_string => tex_get_term_string

    Retrieves a GNUPLOT terminal identifier string.
- procedure, public get_command_string => tex_get_command_string

    Returns the appropriate GNUPLOT command string to establish appropriate parameters.

Private Attributes

- character(len=14) m_id = "epslatex color"

    The terminal ID string.
- character(len=gnuplot_max_path_length) m_fname = "default.tex"

    The filename of the PNG file to write.

6.7.1 Detailed Description

Defines a GNUPLOT LATEX terminal object.

Definition at line 992 of file fplot_core.f90.

6.7.2 Member Function/Subroutine Documentation

6.7.2.1 procedure, public fplot_core::latex_terminal::get_command_string ( )

Returns the appropriate GNUPLOT command string to establish appropriate parameters.

Syntax

```
character(len = :) function, allocatable get_command_string(class(latex_terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The GNUPLOT command string.

Definition at line 1064 of file fplot_core.f90.

6.7.2.2 procedure, public fplot_core::latex_terminal::get_filename ( )

Gets the filename for the output LATEX file.

Syntax

```
character(len = :) function, allocatable get_filename(class(latex_terminal) this)
```

Parameters

in	*this*	The latex_terminal object.

Returns

The filename, including the file extension (.tex).

Example

```
program example
    use fplot_core
    implicit none

    type(latex_terminal) :: term
    character(len = :), allocatable :: fname

    ! Get the filename
    fname = term%get_filename()
end program
```

Definition at line 1021 of file fplot_core.f90.

6.7.2.3 procedure, public fplot_core::latex_terminal::get_id_string ( )

Retrieves a GNUPLOT terminal identifier string.

Syntax

```
character(len = :) function, allocatable get_id_string(class(latex_terminal) this)
```

Parameters

in	*this*	The latex_terminal object.

Returns

The string.

Definition at line 1053 of file fplot_core.f90.

6.7.2.4 procedure, public fplot_core::latex_terminal::set_filename ( )

Sets the filename for the output LATEX file.

Syntax

```
subroutine set_filename(class(latex_terminal) this, character(len = *) txt)
```

Parameters

in,out	*this*	The latex_terminal object.
in	*txt*	The filename, including the file extension (.tex).

Example

```
program example
    use fplot_core
    implicit none

    type(latex_terminal) :: term

    ! Set the filename
    call term%set_filename("Example LATEX File.tex")
end program
```

Definition at line 1043 of file fplot_core.f90.

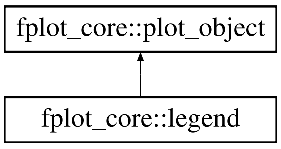
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.8 fplot_core::legend Type Reference

Defines a legend object.

Inheritance diagram for fplot_core::legend:



Public Member Functions

- procedure, public get_draw_inside_axes => leg_get_inside

  Gets a value determining if the legend should be drawn inside the axes border (true), or outside the axes border (false).
- procedure, public set_draw_inside_axes => leg_set_inside

  Sets a value determining if the legend should be drawn inside the axes border (true), or outside the axes border (false).
- procedure, public get_draw_border => leg_get_box

  Gets a value determining if the legend should have a border.
- procedure, public set_draw_border => leg_set_box

  Sets a value determining if the legend should have a border.
- procedure, public get_horizontal_position => leg_get_horz_pos

  Gets the horizontal position of the legend.
- procedure, public set_horizontal_position => leg_set_horz_pos

  Sets the horizontal position of the legend.
- procedure, public get_vertical_position => leg_get_vert_pos

Gets the vertical position of the legend.

- procedure, public set_vertical_position => leg_set_vert_pos

  Gets the vertical position of the legend.
- procedure, public get_is_visible => leg_get_visible

  Gets a value determining if the legend is visible.
- procedure, public set_is_visible => leg_set_visible

  Sets a value determining if the legend is visible.
- procedure, public get_command_string => leg_get_command_txt

  Gets the command string defining the legend properties.

Private Attributes

- logical m_inside = .true.

  Legend on inside or outside of axes.
- logical m_box = .true.

  Draw a box around the legend.
- character(len=20) m_horzposition = LEGEND_RIGHT

  Defines the horizontal position.
- character(len=20) m_vertposition = LEGEND_TOP

  Defines the vertical position.
- logical m_show = .true.

  Determines if the legend is visible.

6.8.1 Detailed Description

Defines a legend object.

Definition at line 1617 of file fplot_core.f90.

6.8.2 Member Function/Subroutine Documentation

6.8.2.1 procedure, public fplot_core::legend::get_command_string ( )

Gets the command string defining the legend properties.

Syntax

```
character(len = :) function, allocatable get_command_string(class(legend) this)
```

Parameters

in	*this*	The legend object.

Returns

The GNUPLOT command string.

Definition at line 1961 of file fplot_core.f90.

6.8.2.2 procedure, public fplot_core::legend::get_draw_border ( )

Gets a value determining if the legend should have a border.

Syntax

```
pure logical function get_draw_border(class(legend) this)
```

Parameters

in	*this*	The legend object.

Returns

The logical value.

Example

```
program example
    use fplot_core
    implicit none

    type(legend) :: leg
    logical :: check

    check = leg%get_draw_border()
end program
```

Definition at line 1753 of file fplot_core.f90.

6.8.2.3 procedure, public fplot_core::legend::get_draw_inside_axes ( )

Gets a value determining if the legend should be drawn inside the axes border (true), or outside the axes border (false).

Syntax

```
pure logical function get_draw_inside_axes(class(legend) this)
```

Parameters

in	*this*	The legend object.

Returns

The logical value.

Example

```
program example
    use fplot_core
    implicit none

    type(legend) :: leg
    logical :: check

    check = leg%get_draw_inside_axes()
end program
```

Definition at line 1653 of file fplot_core.f90.

6.8.2.4 procedure, public fplot_core::legend::get_horizontal_position ( )

Gets the horizontal position of the legend.

Syntax

```
character(len = :) function, allocatable get_horizontal_position(class(legend) this)
```

Parameters

in	*this*	The legend object.

Returns

The horizontal position of the legend (LEGEND_LEFT, LEGEND_CENTER, or LEGEND_RIGHT).

Example

```
program example
    use fplot_core
    implicit none

    type(legend) :: leg
    character(len = :), allocatable :: pos

    pos = leg%get_horizontal_position()
end program
```

Definition at line 1790 of file fplot_core.f90.

6.8.2.5 procedure, public fplot_core::legend::get_is_visible ( )

Gets a value determining if the legend is visible.

Syntax

```
pure logical function get_is_visible(class(legend) this)
```

Parameters

in	*this*	The legend object.

Returns

The logical value.

Example

```
program example
    use fplot_core
    implicit none
    type(legend) :: leg
    logical :: check

    check = leg%get_is_visible()
end program
```

Definition at line 1929 of file fplot_core.f90.

6.8.2.6 procedure, public fplot_core::legend::get_vertical_position ( )

Gets the vertical position of the legend.

Syntax

```
character(len = :) function, allocatable get_vertical_position(class(legend) this)
```

Parameters

in	*this*	The legend object.

Returns

The vertical position of the legend (LEGEND_TOP, LEGEND_CENTER, or LEGEND_BOTTOM).

Example

```
program example
    use fplot_core
    implicit none

    type(legend) :: leg
    character(len = :), allocatable :: pos

    pos = leg%get_vertical_position()
end program
```

Definition at line 1891 of file fplot_core.f90.

6.8.2.7 procedure, public fplot_core::legend::set_draw_border ( )

Sets a value determining if the legend should have a border.

Syntax

```
subroutine set_draw_border(class(legend) this, logical x)
```

Parameters

in,out	*this*	The legend object.
in	*x*	The logical value.

Example

For an example, see set_draw_inside_axes.

Definition at line 1766 of file fplot_core.f90.

6.8.2.8 procedure, public fplot_core::legend::set_draw_inside_axes ( )

Sets a value determining if the legend should be drawn inside the axes border (true), or outside the axes border (false).

Syntax

```
subroutine set_draw_inside_axes(class(legend) this, logical x)
```

Parameters

in,out	*this*	The legend object.
in	*x*	The logical value.

Example

The following example draws a simple plot, adjusts the position of the legend to be located outside the plot axes, and removes the border around the legend.

```
program example
    use iso_fortran_env
    use fplot_core
    implicit none

    ! Local Variables & Parameters
    integer(int32), parameter :: npts = 1000
    real(real64), dimension(npts) :: x, y1, y2
    type(plot_2d) :: plt
    class(terminal), pointer :: term
    type(plot_data_2d) :: d1, d2
    class(plot_axis), pointer :: xaxis, yaxis
    type(legend), pointer :: leg

    ! Build a data set to plot
    x = linspace(0.0d0, 10.0d0, npts)
    y1 = sin(x) * cos(x)
    y2 = sqrt(x) * sin(x)

    call d1%define_data(x, y1)
    call d2%define_data(x, y2)

    ! Set up the plot
    call plt%initialize(gnuplot_terminal_png) ! Save to file directly
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    ! Put the legend outside the axes, and remove it's border
    leg => plt%get_legend()
    call leg%set_draw_inside_axes(.false.)
    call leg%set_draw_border(.false.)

    ! Set up line color and style properties to better distinguish each data set
    call d1%set_name("Data Set 1")
    call d1%set_line_color(clr_blue)

    call d2%set_name("Data Set 2")
    call d2%set_line_color(clr_green)

    ! Add the data to the plot
    call plt%push(d1)
    call plt%push(d2)

    ! Define the file to which the plot should be saved
    term => plt%get_terminal()
    select type (term)
```

```
        class is (png_terminal)
            call term%set_filename("example_plot.png")
        end select

        ! Draw the plot
        call plt%draw()
    end program
```

Definition at line 1730 of file fplot_core.f90.

6.8.2.9 procedure, public fplot_core::legend::set_horizontal_position ( )

Sets the horizontal position of the legend.

Syntax

```
    subroutine set_horizontal_position(class(legend) this, character(len = *) x)
```

Parameters

`in,out`	*this*	The legend object.
	x	The horizontal position of the legend. The parameter must be set to one of the following: LEGEND_LEFT, LEGEND_CENTER, or LEGEND_RIGHT. If not, the default LEGEND_RIGHT will be used.

Example

The following example draws a simple plot, and adjusts the position of the legend.

```
    program example
        use iso_fortran_env
        use fplot_core
        implicit none

        ! Local Variables & Parameters
        integer(int32), parameter :: npts = 1000
        real(real64), dimension(npts) :: x, y1, y2
        type(plot_2d) :: plt
        class(terminal), pointer :: term
        type(plot_data_2d) :: d1, d2
        class(plot_axis), pointer :: xaxis, yaxis
        type(legend), pointer :: leg

        ! Build a data set to plot
        x = linspace(0.0d0, 10.0d0, npts)
        y1 = sin(x) * cos(x)
        y2 = sqrt(x) * sin(x)

        call d1%define_data(x, y1)
        call d2%define_data(x, y2)

        ! Set up the plot
        call plt%initialize(gnuplot_terminal_png) ! Save to file directly
        call plt%set_title("Example Plot")

        xaxis => plt%get_x_axis()
        call xaxis%set_title("X Axis")

        yaxis => plt%get_y_axis()
        call yaxis%set_title("Y Axis")

        ! Put the legend in the upper left corner of the plot
        leg => plt%get_legend()
        call leg%set_horizontal_position(legend_left)
        call leg%set_vertical_position(legend_top)

        ! Set up line color and style properties to better distinguish each data set
        call d1%set_name("Data Set 1")
        call d1%set_line_color(clr_blue)
        call d2%set_name("Data Set 2")
        call d2%set_line_color(clr_green)

        ! Add the data to the plot
        call plt%push(d1)
        call plt%push(d2)

        ! Define the file to which the plot should be saved
        term => plt%get_terminal()
        select type (term)
        class is (png_terminal)
            call term%set_filename("example_plot.png")
        end select

        ! Draw the plot
        call plt%draw()
    end program
```

Definition at line 1867 of file fplot_core.f90.

6.8.2.10 procedure, public fplot_core::legend::set_is_visible ( )

Sets a value determining if the legend is visible.

Syntax

```
    subroutine set_is_visible(class(legend) this, logical x)
```

Parameters

in,out	*this*	The legend object.
in	*x*	The logical value.

Example

```
    program example
        use fplot_core
        implicit none

        type(legend) :: leg

        call leg%set_is_visible(.true.)
    end program
```

Definition at line 1951 of file fplot_core.f90.

6.8.2.11 procedure, public fplot_core::legend::set_vertical_position ( )

Gets the vertical position of the legend.

Syntax

```
    subroutine set_vertical_position(class(legend) this, character(len = *) x)
```

Parameters

in,out	*this*	The legend object.
	x	The vertical position of the legend. The parameter must be set to one of the following: LEGEND_TOP, LEGEND_CENTER, or LEGEND_BOTTOM. If not, the default LEGEND_TOP will be used.

Example

For an example, see set_horizontal_position.

Definition at line 1906 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.9 fplot_core::pa_get_string_result Interface Reference

Retrieves a string from a plot_axis.

Private Member Functions

- character(len=:) function, allocatable **pa_get_string_result** (this)

6.9.1 Detailed Description

Retrieves a string from a plot_axis.

Parameters

`in`	*this*	The plot_axis object.

Returns

The string.

Definition at line 6717 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.10 fplot_core::pd_get_string_result Interface Reference

Retrieves a string from a plot_data object.

Private Member Functions

- character(len=:) function, allocatable **pd_get_string_result** (this)

6.10.1 Detailed Description

Retrieves a string from a plot_data object.

Parameters

in	*this*	The plot_data object.

Returns

The string.

Definition at line 6707 of file fplot_core.f90.

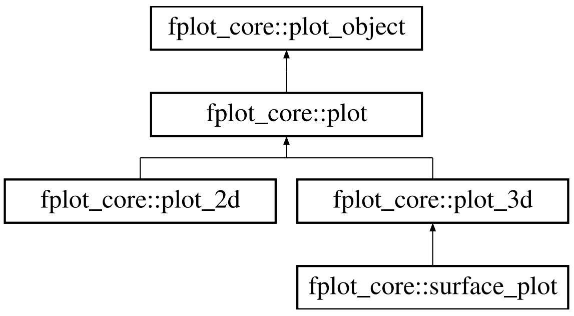
The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.11 fplot_core::plot Type Reference

Defines the basic GNUPLOT plot.

Inheritance diagram for fplot_core::plot:



Public Member Functions

- procedure, public free_resources => plt_clean_up

  Cleans up resources held by the plot object. Inheriting classes are expected to call this routine to free internally held resources.
- procedure, public initialize => plt_init

  Initializes the plot object.
- procedure, public get_title => plt_get_title

  Gets the plot's title.
- procedure, public set_title => plt_set_title

  Sets the plot's title.
- procedure, public is_title_defined => plt_has_title

  Gets a value determining if a title has been defined for the plot object.
- procedure, public get_legend => plt_get_legend

  Gets the plot's legend object.
- procedure, public get_count => plt_get_count

  Gets the number of stored plot_data objects.
- procedure, public push => plt_push_data

  Pushes a plot_data object onto the stack.

- procedure, public pop => plt_pop_data

  Pops the last plot_data object from the stack.
- procedure, public clear_all => plt_clear_all

  Removes all plot_data objects from the plot.
- procedure, public get => plt_get

  Gets a pointer to the requested plot_data object.
- procedure, public set => plt_set

  Sets the requested plot_data object into the plot.
- procedure, public get_terminal => plt_get_term

  Gets the GNUPLOT terminal object.
- procedure, public get_show_gridlines => plt_get_show_grid

  Gets a flag determining if the grid lines should be shown.
- procedure, public set_show_gridlines => plt_set_show_grid

  Sets a flag determining if the grid lines should be shown.
- procedure, public draw => plt_draw

  Launches GNUPLOT and draws the plot per the current state of the command list.
- procedure, public save_file => plt_save

  Saves a GNUPLOT command file.
- procedure, public get_font_name => plt_get_font

  Gets the name of the font used for plot text.
- procedure, public set_font_name => plt_set_font

  Sets the name of the font used for plot text.
- procedure, public get_font_size => plt_get_font_size

  Gets the size of the font used by the plot.
- procedure, public set_font_size => plt_set_font_size

  Sets the size of the font used by the plot.
- procedure, public get_tics_inward => plt_get_tics_in

  Gets a value determining if the axis tic marks should point inwards.
- procedure, public set_tics_inward => plt_set_tics_in

  Sets a value determining if the axis tic marks should point inwards.
- procedure, public get_draw_border => plt_get_draw_border

  Gets a value determining if the border should be drawn.
- procedure, public set_draw_border => plt_set_draw_border

  Sets a value determining if the border should be drawn.

Private Attributes

- character(len=plotdata_max_name_length) m_title = ""

  The plot title.
- logical m_hastitle = .false.

  Has a title?
- class(terminal), pointer m_terminal => null()

  The GNUPLOT terminal object to target.
- type(list) m_data

  A collection of plot_data items to plot.
- type(legend), pointer m_legend => null()

  The legend.
- logical m_showgrid = .true.

  Show grid lines?
- logical m_ticsin = .true.

  Point tic marks in?
- logical m_drawborder = .true.

  Draw the border?

6.11.1 Detailed Description

Defines the basic GNUPLOT plot.

Definition at line 2026 of file fplot_core.f90.

6.11.2 Member Function/Subroutine Documentation

6.11.2.1 procedure, public fplot_core::plot::clear_all ( )

Removes all plot_data objects from the plot.

Syntax

```
subroutine clear(class(plot) this)
```

Parameters

in,out	*this*	The plot object.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_2d) :: plt

    call plt%clear_all()
end program
```

Definition at line 2255 of file fplot_core.f90.

6.11.2.2 procedure, public fplot_core::plot::draw ( )

Launches GNUPLOT and draws the plot per the current state of the command list.

Syntax

```
subroutine draw(class(plot) this, optional logical persist, optional class(errors) err)
```

Parameters

in	*this*	The plot object.
in	*persist*	An optional parameter that can be used to keep GNUPLOT open. Set to true to force GNUPLOT to remain open; else, set to false to allow GNUPLOT to close after drawing. The default is true.
in,out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		• PLOT_GNUPLOT_FILE_ERROR: Occurs if the command file cannot be written.

Example

See `png_terminal` for an example.

Definition at line 2398 of file fplot_core.f90.

6.11.2.3 procedure, public fplot_core::plot::free_resources ( )

Cleans up resources held by the plot object. Inheriting classes are expected to call this routine to free internally held resources.

Syntax

```
module free_resources(class(plot) this)
```

Parameters

in,out	*this*	The plot object.

Definition at line 2055 of file fplot_core.f90.

6.11.2.4 procedure, public fplot_core::plot::get ( )

Gets a pointer to the requested plot_data object.

Syntax

```
class(plot_data) function, pointer get(class(plot), integer(int32) i)
```

Parameters

in	*this*	The plot object.
in	*i*	The index of the plot_data object.

Returns

A pointer to the requested plot_data object.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    class(plot_data), pointer :: ptr

    ! Add some data ... (not shown)

    ! Retrieve the second data set added
    ptr => plt%get(2)
end program
```

Definition at line 2284 of file fplot_core.f90.

6.11.2.5 procedure, public fplot_core::plot::get_count ( )

Gets the number of stored plot_data objects.

Syntax

```
pure integer(int32) function get_count(class(plot) this)
```

Parameters

in	*this*	The plot object.

Returns

The number of plot_data objects.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_2d) :: plt
    integer(int32) :: n

    n = plt%get_count()
end program
```

Definition at line 2189 of file fplot_core.f90.

6.11.2.6 procedure, public fplot_core::plot::get_draw_border ( )

Gets a value determining if the border should be drawn.

Syntax

```
pure logical function get_draw_border(class(plot) this)
```

Parameters

in	*this*	The plot object.

Returns

Returns true if the border should be drawn; else, false.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    logical :: check

    check = plt%get_draw_border()
end program
```

Definition at line 2642 of file fplot_core.f90.

6.11.2.7 procedure, public fplot_core::plot::get_font_name ( )

Gets the name of the font used for plot text.

Syntax

```
character(len = :) function, allocatable get_font_name(class(plot) this)
```

Parameters

in	*this*	The plot object.

Returns

The font name.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    character(len = :), allocatable :: name

    name = plt%get_font_name()
end program
```

Definition at line 2485 of file fplot_core.f90.

6.11.2.8 procedure, public fplot_core::plot::get_font_size ( )

Gets the size of the font used by the plot.

Syntax

```
integer(int32) function get_font_size(class(plot) this)
```

Parameters

in	*this*	The plot object.

Returns

The size of the font, in points.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_2d) :: plt
    integer(int32) :: sz

    sz = plt%get_font_size()
end program
```

Definition at line 2536 of file fplot_core.f90.

6.11.2.9 procedure, public fplot_core::plot::get_legend ( )

Gets the plot's legend object.

Syntax

```
class(legend) function, pointer get_legend(class(this) plot)
```

Parameters

in	*this*	The plot object.

Returns

A pointer to the legend object.

Example

See `png_terminal` for an example.

Example

See `png_terminal` for an example.

Definition at line 2163 of file fplot_core.f90.

6.11.2.10 procedure, public fplot_core::plot::get_show_gridlines ( )

Gets a flag determining if the grid lines should be shown.

Syntax

```
pure logical function get_show_gridlines(class(plot) this)
```

Parameters

in	*this*	The plot object.

Returns

Returns true if the grid lines should be shown; else, false.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    logical :: check

    check = plt%get_show_gridlines()
end program
```

Definition at line 2351 of file fplot_core.f90.

6.11.2.11 procedure, public fplot_core::plot::get_terminal ( )

Gets the GNUPLOT terminal object.

Syntax

```
class(terminal) function, pointer get_terminal(class(plot) this)
```

Parameters

in	*this*	The plot object.

Returns

A pointer to the GNUPLOT terminal object.

Example

See `png_terminal` for an example.

Definition at line 2326 of file fplot_core.f90.

6.11.2.12 procedure, public fplot_core::plot::get_tics_inward ( )

Gets a value determining if the axis tic marks should point inwards.

Syntax

```
pure logical function get_tics_inward(class(plot) this)
```

Parameters

in	*this*	The plot object.

Returns

Returns true if the tic marks should point inwards; else, false if the tic marks should point outwards.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    logical :: check

    check = plt%get_tics_inward()
end program
```

Definition at line 2590 of file fplot_core.f90.

6.11.2.13 procedure, public fplot_core::plot::get_title ( )

Gets the plot's title.

Syntax

```
character(len = :) function, allocatable get_title(class(plot))
```

Parameters

in	*this*	The plot object.

Returns

The plot's title.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    character(len = :), allocatable :: txt

    txt = plt%get_title()
end program
```

Definition at line 2105 of file fplot_core.f90.

6.11.2.14 procedure, public fplot_core::plot::initialize ( )

Initializes the plot object.

Syntax

```
subroutine initialize(class(plot) this, optional class(terminal) term, optional class(errors) err)
```

Parameters

`in,out`	*this*	The plot object.
`in`	*term*	An optional input that is used to define the terminal. The default terminal is a WXT terminal. The acceptable inputs are: • GNUPLOT_TERMINAL_PNG • GNUPLOT_TERMINAL_QT • GNUPLOT_TERMINAL_WIN32 • GNUPLOT_TERMINAL_WXT • GNUPLOT_TERMINAL_LATEX
`in,out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Example

See `png_terminal` for an example.

Definition at line 2080 of file fplot_core.f90.

6.11.2.15 procedure, public fplot_core::plot::is_title_defined ( )

Gets a value determining if a title has been defined for the plot object.

Syntax

```
pure logical function is_title_defined(class(plot) this)
```

Parameters

`in`	*this*	The plot object.

Returns

Returns true if a title has been defined for this plot; else, returns false.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    logical :: check

    check = plt%is_title_defined()
end program
```

Definition at line 2147 of file fplot_core.f90.

6.11.2.16 procedure, public fplot_core::plot::pop ( )

Pops the last plot_data object from the stack.

Syntax

```
subroutine pop(class(plot) this)
```

Parameters

in,out	*this*	The plot object.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt

    call plt%pop()
end program
```

Definition at line 2231 of file fplot_core.f90.

6.11.2.17 procedure, public fplot_core::plot::push ( )

Pushes a plot_data object onto the stack.

Syntax

```
subroutine push(class(plot) this, class(plot_data) x, optional class(errors) err)
```

Parameters

in,out	*this*	The plot object.
in	*x*	The plot_data object.
in,out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		• PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Example

See `png_terminal` for an example.

Definition at line 2208 of file fplot_core.f90.

6.11.2.18 procedure, public fplot_core::plot::save_file ( )

Saves a GNUPLOT command file.

Syntax

```
subroutine save_file(class(plot) this, character(len = *) fname, optional class(errors) err)
```

Parameters

`in`	*this*	The plot object.
`in`	*fname*	The filename.
`in,out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_GNUPLOT_FILE_ERROR: Occurs if the command file cannot be written.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    ! Local Variables & Parameters
    integer(int32), parameter :: npts = 1000
    real(real64), dimension(npts) :: x, y
    type(plot_2d) :: plt
    type(plot_data_2d) :: dataset
    class(plot_axis), pointer :: xaxis, yaxis
    type(legend), pointer :: leg

    ! Build a data set to plot
    x = linspace(0.0d0, 10.0d0, npts)
    y = exp(-0.5d0 * x) * sin(10.0d0 * x - 0.5d0)

    call dataset%define_data(x, y)

    ! Set up the plot
    call plt%initialize()
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    ! Hide the legend
    leg => plt%get_legend()
    call leg%set_is_visible(.false.)

    ! Add the data to the plot
    call plt%push(dataset)

    ! Save the plot to a file that can be opened by GNUPLOT at a later time
    call plt%save_file("example_gnuplot_file.plt")
end program
```

Then, from gnuplot, simply issue the command: load "example_gnuplot_file.plt" to obtain the plot.

Definition at line 2460 of file fplot_core.f90.

6.11.2.19 procedure, public fplot_core::plot::set ( )

Sets the requested plot_data object into the plot.

Syntax

```
subroutine set(class(plot) this, integer(int32) i, class(plot_data) x)
```

Parameters

in,out	*this*	The plot object.
in	*i*	The index of the plot_data object.
in	*x*	The plot_data object.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    type(plot_data_2d) :: dataset

    ! Add some data to the plot ... (not shown)

    ! Add dataset to the second spot in the collection
    call plt%set(2, dataset)
end program
```

Definition at line 2313 of file fplot_core.f90.

6.11.2.20 procedure, public fplot_core::plot::set_draw_border ( )

Sets a value determining if the border should be drawn.

Syntax

```
subroutine set_draw_border(class(plot) this, logical x)
```

Parameters

in,out	*this*	The plot object.
in	*x*	Set to true if the border should be drawn; else, false.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none
    type(plot_2d) :: plt

    ! Shut off the axes border
    call plt%set_draw_border(.false.)
end program
```

Definition at line 2667 of file fplot_core.f90.

6.11.2.21 procedure, public fplot_core::plot::set_font_name ( )

Sets the name of the font used for plot text.

Syntax

```
subroutine set_font_name(class(plot) this, character(len = *) x)
```

Parameters

in,out	*this*	The plot object.
in	*x*	The font name.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt

    ! Establish the font used by the plot as Arial.
    call plt%set_title("Arial")
end program
```

Definition at line 2510 of file fplot_core.f90.

6.11.2.22 procedure, public fplot_core::plot::set_font_size ( )

Sets the size of the font used by the plot.

Syntax

```
subroutine set_font_size(class(plot) this, integer(int32) x)
```

Parameters

in,out	*this*	The plot object.
in	*x*	The font size, in points. If a value of zero is provided, the font size is reset to its default value; or, if a negative value is provided, the absolute value of the supplied value is utilized.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt

    ! Set the font to be 14 point in size
    call plt%set_font_size(14)
end program
```

Definition at line 2563 of file fplot_core.f90.

6.11.2.23 procedure, public fplot_core::plot::set_show_gridlines ( )

Sets a flag determining if the grid lines should be shown.

Syntax

```
subroutine set_show_gridlines(class(plot) this, logical x)
```

Parameters

in,out	*this*	The plot object.
in	*x*	Set to true if the grid lines should be shown; else, false.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt

    ! Turn off the gridlines
    call plt%set_show_gridlines(.false.)
end program
```

Definition at line 2376 of file fplot_core.f90.

6.11.2.24 procedure, public fplot_core::plot::set_tics_inward ( )

Sets a value determining if the axis tic marks should point inwards.

Syntax

```
subroutine set_tics_inward(class(plot) this, logical x)
```

Parameters

in,out	*this*	The plot object.
in	*x*	Set to true if the tic marks should point inwards; else, false if the tic marks should point outwards.

Example

This example uses a plot_2d type, but this example is valid for any type that derives from the plot type.

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt

    ! Point the axes tic marks outward
    call plt%set_tics_inward(.false.)
end program
```

Definition at line 2617 of file fplot_core.f90.

6.11.2.25 procedure, public fplot_core::plot::set_title ( )

Sets the plot's title.

Syntax

```
subroutine set_title(class(plot) this, character(len = *) txt)
```

Parameters

in,out	*this*	The plot object.
in	*txt*	The plot's title. The number of characters must be less than or equal to PLOTDATA_MAX_NAME_LENGTH; else, the text string is truncated.

Example

See `png_terminal` for an example.

Definition at line 2120 of file fplot_core.f90.

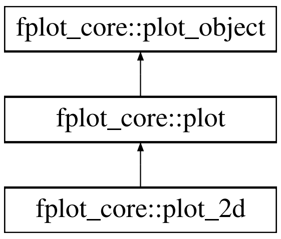
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.12 fplot_core::plot_2d Type Reference

A plot object defining a 2D plot.

Inheritance diagram for fplot_core::plot_2d:



Public Member Functions

- procedure, public initialize => p2d_init

  Initializes the plot_2d object.
- procedure, public get_command_string => p2d_get_cmd

  Gets the GNUPLOT command string to represent this plot_2d object.
- procedure, public get_x_axis => p2d_get_x_axis

  Gets the x-axis object.
- procedure, public get_y_axis => p2d_get_y_axis

  Gets the y-axis object.
- procedure, public get_y2_axis => p2d_get_y2_axis

  Gets the secondary y-axis object.
- procedure, public get_use_y2_axis => p2d_get_use_y2

  Gets a flag determining if the secondary y-axis should be displayed.
- procedure, public set_use_y2_axis => p2d_set_use_y2

  Sets a flag determining if the secondary y-axis should be displayed.

Private Member Functions

- final p2d_clean_up

  Cleans up resources held by the plot_2d object.

Private Attributes

- type(x_axis), pointer m_xaxis => null()

  The x-axis.
- type(y_axis), pointer m_yaxis => null()

  The y-axis.
- type(y2_axis), pointer m_y2axis => null()

  The secondary y-axis.
- logical m_usey2 = .false.

  Display the secondary y axis?

6.12.1 Detailed Description

A plot object defining a 2D plot.

Example

The following example illustrates a 2D plot, and several examples of how to modify various plot settings.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: n = 1000

    ! Local Variables
    real(real64), dimension(n) :: x, y1, y2
    type(plot_2d) :: plt
    type(plot_data_2d) :: d1, d2
    class(plot_axis), pointer :: xaxis, yaxis
    type(legend), pointer :: leg
```

```
        ! Initialize the plot object
        call plt%initialize()

        ! Define titles
        call plt%set_title("2D Example Plot 1")
        call plt%set_font_size(14)

        xaxis => plt%get_x_axis()
        call xaxis%set_title("X Axis")

        yaxis => plt%get_y_axis()
        call yaxis%set_title("Y Axis")

        ! Establish legend properties
        leg => plt%get_legend()
        call leg%set_draw_inside_axes(.false.)
        call leg%set_horizontal_position(legend_center)
        call leg%set_vertical_position(legend_bottom)
        call leg%set_draw_border(.false.)

        ! Define the data, and then add it to the plot
        x = linspace(0.0d0, 10.0d0, n)
        y1 = sin(5.0d0 * x)
        y2 = 2.0d0 * cos(2.0d0 * x)

        call d1%define_data(x, y1)
        call d2%define_data(x, y2)

        ! Define properties for each data set
        call d1%set_name("Data Set 1")
        call d1%set_line_color(clr_blue)
        call d1%set_draw_markers(.true.)
        call d1%set_marker_frequency(10)
        call d1%set_marker_style(marker_empty_circle)
        call d1%set_marker_scaling(2.0)

        call d2%set_name("Data Set 2")
        call d2%set_line_color(clr_green)
        call d2%set_line_style(line_dashed)
        call d2%set_line_width(2.0)

        ! Add the data sets to the plot
        call plt%push(d1)
        call plt%push(d2)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 5089 of file fplot_core.f90.

6.12.2 Member Function/Subroutine Documentation

6.12.2.1 procedure, public fplot_core::plot_2d::get_command_string ( )

Gets the GNUPLOT command string to represent this plot_2d object.

Syntax

```
    character(len = :) function, allocatable get_command_string(class(plot_2d) this)
```

Parameters

in	*this*	The plot_2d object.

Returns

The command string.

Definition at line 5144 of file fplot_core.f90.

6.12.2.2 procedure, public fplot_core::plot_2d::get_use_y2_axis ( )

Gets a flag determining if the secondary y-axis should be displayed.

Syntax

```
pure logical function get_use_y2_axis(class(plot_2d) this)
```

Parameters

in	*this*	The plot_2d object.

Returns

Returns true if the axis should be displayed; else, false.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    logical :: check

    ! Determine if a secondary y axis is in use
    check = plt%get_use_y2_axis()
end program
```

Definition at line 5241 of file fplot_core.f90.

6.12.2.3 procedure, public fplot_core::plot_2d::get_x_axis ( )

Gets the x-axis object.

Syntax

```
class(plot_axis) function, pointer get_x_axis(class(plot_2d) this)
```

Parameters

in	*this*	The plot_2d object.

Returns

A pointer to the x-axis object.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    class(plot_axis) :: axis
```

```
        ! Get a pointer to the axis object
        axis => plt%get_x_axis()
    end program
```

Definition at line 5168 of file fplot_core.f90.

6.12.2.4 procedure, public fplot_core::plot_2d::get_y2_axis ( )

Gets the secondary y-axis object.

Syntax

```
    class(plot_axis) function, pointer get_y2_axis(class(plot_2d) this)
```

Parameters

in	*this*	The plot_2d object.

Returns

A pointer to the secondary y-axis object.

Example

```
    program example
        use fplot_core
        implicit none

        type(plot_2d) :: plt
        class(plot_axis) :: axis

        ! Get a pointer to the axis object
        axis => plt%get_y2_axis()
    end program
```

Definition at line 5216 of file fplot_core.f90.

6.12.2.5 procedure, public fplot_core::plot_2d::get_y_axis ( )

Gets the y-axis object.

Syntax

```
    class(plot_axis) function, pointer get_y_axis(class(plot_2d) this)
```

Parameters

in	*this*	The plot_2d object.

Returns

A pointer to the y-axis object.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_2d) :: plt
    class(plot_axis) :: axis

    ! Get a pointer to the axis object
    axis => plt%get_y_axis()
end program
```

Definition at line 5192 of file fplot_core.f90.

6.12.2.6 procedure, public fplot_core::plot_2d::initialize ( )

Initializes the plot_2d object.

Syntax

```
subroutine initialize(class(plot_2d) this, optional integer(int32) term, optional class(errors) err)
```

Parameters

`in`	*this*	The plot_2d object.
`in`	*term*	An optional input that is used to define the terminal. The default terminal is a WXT terminal. The acceptable inputs are: • GNUPLOT_TERMINAL_PNG • GNUPLOT_TERMINAL_QT • GNUPLOT_TERMINAL_WIN32 • GNUPLOT_TERMINAL_WXT • GNUPLOT_TERMINAL_LATEX
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Example

See png_terminal for an example.

Definition at line 5133 of file fplot_core.f90.

6.12.2.7 final fplot_core::plot_2d::p2d_clean_up ( ) `[final]`, `[private]`

Cleans up resources held by the plot_2d object.

Syntax

```
subroutine p2d_clean_up(type(plot_2d) this)
```

Parameters

in,out	*this*	The plot_2d object.

Definition at line 5108 of file fplot_core.f90.

6.12.2.8 procedure, public fplot_core::plot_2d::set_use_y2_axis ( )

Sets a flag determining if the secondary y-axis should be displayed.

Syntax

```
subroutine set_use_y2_axis(class(plot_2d) this, logical x)
```

Parameters

in,out	*this*	The plot_2d object.
in	*x*	Set to true if the axis should be displayed; else, false.

Example

This example illustrates the use of a secondary y axis.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    ! Local Variables
    integer(int32), parameter :: npts = 1000
    real(real64), dimension(npts) :: x, y1, y2
    type(plot_2d) :: plt
    type(plot_data_2d) :: ds1, ds2
    class(plot_axis), pointer :: xaxis, yaxis, y2axis

    ! Build a data set
    x = linspace(0.0d0, 10.0d0, npts)
    y1 = exp(-0.5d0 * x) * abs(sin(x))
    y2 = cos(0.5d0 * x) * sin(10.0d0 * x)

    call ds1%define_data(x, y1)
    call ds1%set_name("f(x) = exp(-x / 2) * |sin(x)|")

    call ds2%define_data(x, y2)
    call ds2%set_name("f(x) = cos(x / 2) * sin(10 x)")

    ! Make the ds2 line green and dashed
    call ds2%set_line_color(clr_green)
    call ds2%set_line_style(line_dashed)

    ! Draw ds2 against the secondary y axis
    call ds2%set_draw_against_y2(.true.)

    ! Ensure the plot knows it needs a secondary y axis
    call plt%set_use_y2_axis(.true.)

    ! Set up the plot
    call plt%initialize()
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    y2axis => plt%get_y2_axis()
    call y2axis%set_title("Secondary Y Axis")
```

```
        ! Add the data to the plot
        call plt%push(ds1)
        call plt%push(ds2)

        ! Draw
        call plt%draw()
    end program
```

Definition at line 5311 of file fplot_core.f90.

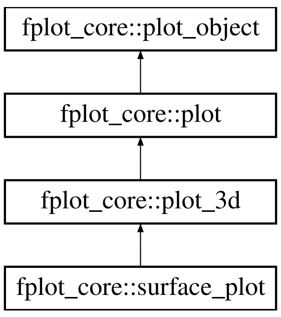
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.13 fplot_core::plot_3d Type Reference

A plot object defining a 3D plot.

Inheritance diagram for fplot_core::plot_3d:



Public Member Functions

- procedure, public initialize => p3d_init

  Initializes the plot_3d object.
- procedure, public get_command_string => p3d_get_cmd

  Gets the GNUPLOT command string to represent this plot_3d object.
- procedure, public get_x_axis => p3d_get_x_axis

  Gets the x-axis object.
- procedure, public get_y_axis => p3d_get_y_axis

  Gets the y-axis object.
- procedure, public get_z_axis => p3d_get_z_axis

  Gets the z-axis object.
- procedure, public get_elevation => p3d_get_elevation

  Gets the plot elevation angle.
- procedure, public set_elevation => p3d_set_elevation

  Sets the plot elevation angle.
- procedure, public get_azimuth => p3d_get_azimuth

  Gets the plot azimuth angle.
- procedure, public set_azimuth => p3d_set_azimuth

  Sets the plot azimuth angle.
- procedure, public get_z_intersect_xy => p3d_get_z_axis_intersect

  Gets a value determining if the z-axis should intersect the x-y plane.
- procedure, public set_z_intersect_xy => p3d_set_z_axis_intersect

  Sets a value determining if the z-axis should intersect the x-y plane.

Private Member Functions

- final p3d_clean_up

  Cleans up resources held by the plot_3d object.

Private Attributes

- type(x_axis), pointer m_xaxis => null()

  The x-axis.
- type(y_axis), pointer m_yaxis => null()

  The y-axis.
- type(z_axis), pointer m_zaxis => null()

  The z-axis.
- real(real64) m_elevation = 60.0d0

  The elevation angle.
- real(real64) m_azimuth = 30.0d0

  The azimuth.
- logical m_zintersect = .true.

  Z-axis intersect X-Y plane?

6.13.1 Detailed Description

A plot object defining a 3D plot.

Example

The following example adds data to draw a helix to a 3D plot.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: n = 1000

    ! Local Variables
    real(real64), dimension(n) :: t, x, y, z
    type(plot_3d) :: plt
    type(plot_data_3d) :: d1
    class(plot_axis), pointer :: xaxis, yaxis, zaxis
    type(legend), pointer :: leg

    ! Initialize the plot object
    call plt%initialize()
    leg => plt%get_legend()
    call leg%set_is_visible(.false.)

    ! Define titles
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    zaxis => plt%get_z_axis()
    call zaxis%set_title("Z Axis")

    ! Define the data
    t = linspace(0.0d0, 10.0d0, n)
    x = cos(5.0d0 * t)
    y = sin(5.0d0 * t)
    z = 2.0d0 * t
```

```
        call d1%define_data(x, y, z)

        ! Set up the data set
        call d1%set_line_color(clr_blue)
        call d1%set_line_width(2.0)

        ! Add the data to the plot
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 5417 of file fplot_core.f90.

6.13.2 Member Function/Subroutine Documentation

6.13.2.1 procedure, public fplot_core::plot_3d::get_azimuth ( )

Gets the plot azimuth angle.

Syntax

```
    real(real64) function get_azimuth(class(plot_3d) this)
```

Parameters

in	*this*	The plot_3d object.

Returns

The azimuth angle, in degrees.

Example

```
    program example
        use fplot_core
        use iso_fortran_env
        implicit none

        type(plot_3d) :: plt
        real(real64) :: val

        ! Get the azimuth angle of the plot
        val = plt%get_azimuth()
    end program
```

Definition at line 5614 of file fplot_core.f90.

6.13.2.2 procedure, public fplot_core::plot_3d::get_command_string ( )

Gets the GNUPLOT command string to represent this plot_3d object.

Syntax

```
    character(len = :) function, allocatable get_command_string(class(plot_3d) this)
```

Parameters

in	*this*	The plot_3d object.

Returns

The command string.

Definition at line 5468 of file fplot_core.f90.

6.13.2.3 procedure, public fplot_core::plot_3d::get_elevation ( )

Gets the plot elevation angle.

Syntax

```
real(real64) function get_elevation(class(plot_3d) this)
```

Parameters

in	*this*	The plot_3d object.

Returns

The elevation angle, in degrees.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_3d) :: plt
    real(real64) :: val

    ! Get the elevation angle of the plot
    val = plt%get_elevation()
end program
```

Definition at line 5565 of file fplot_core.f90.

6.13.2.4 procedure, public fplot_core::plot_3d::get_x_axis ( )

Gets the x-axis object.

Syntax

```
class(plot_axis) function, pointer get_x_axis(class(plot_3d) this)
```

Parameters

in	*this*	The plot_3d object.

Returns

A pointer to the x-axis object.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_3d) :: plt
    class(plot_axis) :: axis

    ! Get a pointer to the axis object
    axis => plt%get_x_axis()
end program
```

Definition at line 5492 of file fplot_core.f90.

6.13.2.5 procedure, public fplot_core::plot_3d::get_y_axis ( )

Gets the y-axis object.

Syntax

```
class(plot_axis) function, pointer get_y_axis(class(plot_3d) this)
```

Parameters

in	*this*	The plot_3d object.

Returns

A pointer to the y-axis object.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_3d) :: plt
    class(plot_axis) :: axis

    ! Get a pointer to the axis object
    axis => plt%get_y_axis()
end program
```

Definition at line 5516 of file fplot_core.f90.

6.13.2.6 procedure, public fplot_core::plot_3d::get_z_axis ( )

Gets the z-axis object.

Syntax

```
class(plot_axis) function, pointer get_z_axis(class(plot_3d) this)
```

Parameters

in	*this*	The plot_3d object.

Returns

A pointer to the z-axis object.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_3d) :: plt
    class(plot_axis) :: axis

    ! Get a pointer to the axis object
    axis => plt%get_z_axis()
end program
```

Definition at line 5540 of file fplot_core.f90.

6.13.2.7 procedure, public fplot_core::plot_3d::get_z_intersect_xy ( )

Gets a value determining if the z-axis should intersect the x-y plane.

Syntax

```
pure logical function get_z_intersect_xy(class(plot_3d) this)
```

Parameters

in	*this*	The plot_3d object.

Returns

Returns true if the z-axis should intersect the x-y plane; else, false to allow the z-axis to float.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_3d) :: plt
    logical :: check

    ! Determine if the z axis is drawn to intersect the x-y plane
    check = plt%get_z_intersect_xy()
end program
```

Definition at line 5664 of file fplot_core.f90.

6.13.2.8 procedure, public fplot_core::plot_3d::initialize ( )

Initializes the plot_3d object.

Syntax

```
subroutine initialize(class(plot_3d) this, optional integer(int32) term, optional class(errors) err)
```

Parameters

in	*this*	The plot_3d object.
in	*term*	An optional input that is used to define the terminal. The default terminal is a WXT terminal. The acceptable inputs are: • GNUPLOT_TERMINAL_PNG • GNUPLOT_TERMINAL_QT • GNUPLOT_TERMINAL_WIN32 • GNUPLOT_TERMINAL_WXT • GNUPLOT_TERMINAL_LATEX
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Definition at line 5457 of file fplot_core.f90.

6.13.2.9 final fplot_core::plot_3d::p3d_clean_up ( ) [final], [private]

Cleans up resources held by the plot_3d object.

Parameters

in,out	*this*	The plot_3d object.

Definition at line 5435 of file fplot_core.f90.

6.13.2.10 procedure, public fplot_core::plot_3d::set_azimuth ( )

Sets the plot azimuth angle.

Syntax

```
subroutine set_azimuth(class(plot_3d) this, real(real64) x)
```

Parameters

in,out	*this*	The plot_3d object.
in	*x*	The azimuth angle, in degrees.

Example

```
program example
    use fplot_core
    use iso_fortran_env
```

```
        implicit none

        type(plot_3d) :: plt

        ! Set the azimuth angle of the plot
        call plt%set_azimuth(15.0d0)
    end program
```

Definition at line 5638 of file fplot_core.f90.

6.13.2.11 procedure, public fplot_core::plot_3d::set_elevation ( )

Sets the plot elevation angle.

Syntax

```
    subroutine set_elevation(class(plot_3d) this, real(real64) x)
```

Parameters

in,out	*this*	The plot_3d object.
in	*x*	The elevation angle, in degrees.

Example

```
    program example
        use fplot_core
        use iso_fortran_env
        implicit none

        type(plot_3d) :: plt

        ! Set the elevation angle of the plot
        call plt%set_elevation(15.0d0)
    end program
```

Definition at line 5589 of file fplot_core.f90.

6.13.2.12 procedure, public fplot_core::plot_3d::set_z_intersect_xy ( )

Sets a value determining if the z-axis should intersect the x-y plane.

Syntax

```
    subroutine set_z_intersect_xy(class(plot_3d) this, logical x)
```

Parameters

in,out	*this*	The plot_3d object.
in	*x*	Set to true if the z-axis should intersect the x-y plane; else, false to allow the z-axis to float.

Example

```
    program example
```

```
        use, intrinsic :: iso_fortran_env
        use fplot_core
        implicit none

        ! Parameters
        integer(int32), parameter :: n = 1000

        ! Local Variables
        real(real64), dimension(n) :: t, x, y, z
        type(plot_3d) :: plt
        type(plot_data_3d) :: d1
        class(plot_axis), pointer :: xaxis, yaxis, zaxis
        type(legend), pointer :: leg

        ! Initialize the plot object
        call plt%initialize()
        leg => plt%get_legend()
        call leg%set_is_visible(.false.)

        ! Set the Z-axis to not intersect the X-Y plane
        call plt%set_z_intersect_xy(.false.)

        ! Define titles
        call plt%set_title("Example Plot")

        xaxis => plt%get_x_axis()
        call xaxis%set_title("X Axis")

        yaxis => plt%get_y_axis()
        call yaxis%set_title("Y Axis")

        zaxis => plt%get_z_axis()
        call zaxis%set_title("Z Axis")

        ! Define the data
        t = linspace(0.0d0, 10.0d0, n)
        x = cos(5.0d0 * t)
        y = sin(5.0d0 * t)
        z = 2.0d0 * t

        call d1%define_data(x, y, z)

        ! Set up the data set
        call d1%set_line_color(clr_blue)
        call d1%set_line_width(2.0)

        ! Add the data to the plot
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

The above code results in the following plot.

Compare to the default (allowing the z-axis to intersect the x-y plane).

Definition at line 5742 of file fplot_core.f90.

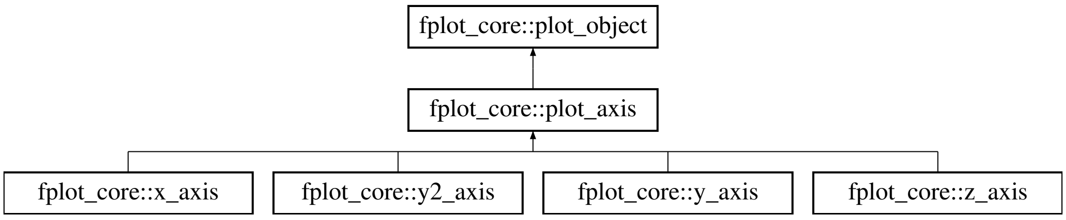
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.14 fplot_core::plot_axis Type Reference

Describes a single plot axis.

Inheritance diagram for fplot_core::plot_axis:



Public Member Functions

- procedure, public get_title => pa_get_title

  Gets the axis' title.
- procedure, public set_title => pa_set_title

  Sets the axis' title.
- procedure, public is_title_defined => pa_has_title

  Gets a value determining if a title has been defined for the plot_axis object.
- procedure, public get_autoscale => pa_get_autoscale

  Gets a logical value determining if the axis should be automatically scaled to fit the data.
- procedure, public set_autoscale => pa_set_autoscale

  Sets a logical value determining if the axis should be automatically scaled to fit the data.
- procedure, public get_limits => pa_get_axis_limits

  Gets the axis display limits, assuming autoscaling is not active for this axis.
- procedure, public set_limits => pa_set_axis_limits

  Sets the axis display limits, assuming autoscaling is not active for this axis.
- procedure, public get_is_log_scaled => pa_get_log_scale

  Gets a logical value defining if the axis should be log scaled.
- procedure, public set_is_log_scaled => pa_set_log_scale

  Sets a logical value defining if the axis should be log scaled.
- procedure, public get_command_string => pa_get_cmd_string

  Returns the appropriate GNUPLOT command string to define the plot_axis properties.
- procedure, public get_zero_axis => pa_get_zero_axis

  Gets a value determining if the axis should be drawn through zero of opposing axes.
- procedure, public set_zero_axis => pa_set_zero_axis

  Sets a value determining if the axis should be drawn through zero of opposing axes.
- procedure, public get_zero_axis_line_width => pa_get_zero_axis_width

  Gets the width of the line used to represent the zero axis line, if active.
- procedure, public set_zero_axis_line_width => pa_set_zero_axis_width

  Sets the width of the line used to represent the zero axis line, if active.
- procedure(pa_get_string_result), deferred, public get_id_string

  Gets a string identifying the axis as: x, y, z, y2, etc.

Private Attributes

- logical m_hastitle = .false.

  Has a title.
- character(len=plotdata_max_name_length) m_title = ""

  The axis title.
- logical m_autoscale = .true.

  Autoscale?
- real(real64), dimension(2) m_limits = [0.0d0, 1.0d0]

Display limits.
- logical m_logscale = .false.

  Log scaled?
- logical m_zeroaxis = .false.

  Zero axis?
- real(real32) m_axiswidth = 1.0

  The width, in pixels, of the zero axis line.

6.14.1 Detailed Description

Describes a single plot axis.

Definition at line 1166 of file fplot_core.f90.

6.14.2 Member Function/Subroutine Documentation

6.14.2.1 procedure, public fplot_core::plot_axis::get_autoscale ( )

Gets a logical value determining if the axis should be automatically scaled to fit the data.

Syntax

```
pure logical function get_autoscale(class(plot_axis) this)
```

Parameters

in	*this*	The plot_axis object.

Returns

Returns true if the axis should be automatically scaled; else, false.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis
    logical :: check

    check = axis%get_autoscale()
end program
```

Definition at line 1287 of file fplot_core.f90.

6.14.2.2 procedure, public fplot_core::plot_axis::get_command_string ( )

Returns the appropriate GNUPLOT command string to define the plot_axis properties.

Syntax

```
character(len = :) function, allocatable get_command_string(class(plot_axis) this)
```

Parameters

in	*this*	The plot_axis object.

Returns

The GNUPLOT command string.

Definition at line 1431 of file fplot_core.f90.

6.14.2.3 procedure, public fplot_core::plot_axis::get_is_log_scaled ( )

Gets a logical value defining if the axis should be log scaled.

Syntax

```
pure logical function get_is_log_scaled(class(plot_axis) this)
```

Parameters

in,out	*this*	The plot_axis object.

Returns

Returns true if log scaling is applied to the axis; else, false.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis
    logical :: check

    check = axis%get_is_log_scaled()
end program
```

Definition at line 1394 of file fplot_core.f90.

6.14.2.4 procedure, public fplot_core::plot_axis::get_limits ( )

Gets the axis display limits, assuming autoscaling is not active for this axis.

Syntax

```
pure real(real64) function, dimension(2) get_limits(class(plot_axis) this)
```

Parameters

in	*this*	The plot_axis object.

Returns

A two-element array containing the limits as follows: [lower, upper].

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(x_axis) :: axis
    real(real64) :: lim(2)

    lim = axis%get_limits()
end program
```

Definition at line 1341 of file fplot_core.f90.

6.14.2.5 procedure, public fplot_core::plot_axis::get_title ( )

Gets the axis' title.

Syntax

```
character(len = :) function, allocatable get_title(class(plot_axis) this)
```

Parameters

in	*this*	The plot_axis object.

Returns

The title.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis
    character(len = :), allocatable :: txt

    txt = axis%get_title()
end program
```

Definition at line 1207 of file fplot_core.f90.

6.14.2.6 procedure, public fplot_core::plot_axis::get_zero_axis ( )

Gets a value determining if the axis should be drawn through zero of opposing axes.

Syntax

```
pure logical function get_zero_axis(class(plot_axis) this)
```

Parameters

in	*this*	The plot_axis object.

Returns

Returns true to draw as a zero axis; else, set to false.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis
    logical :: check

    check = axis%get_zero_axis()
end program
```

Definition at line 1457 of file fplot_core.f90.

6.14.2.7 procedure, public fplot_core::plot_axis::get_zero_axis_line_width ( )

Gets the width of the line used to represent the zero axis line, if active.

Syntax

```
pure real(real32) function get_zero_axis_line_width(class(plot_axis) this)
```

Parameters

in	*this*	The plot_axis object.

Returns

The width of the line, in pixels.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(x_axis) :: axis
    real(real32) :: width

    width = axis%get_zero_axis_line_width()
end program
```

Definition at line 1509 of file fplot_core.f90.

6.14.2.8 procedure, public fplot_core::plot_axis::is_title_defined ( )

Gets a value determining if a title has been defined for the plot_axis object.

Syntax

```
pure logical function is_title_defined(class(plot_axis) this)
```

Parameters

in	*this*	The plot_axis object.

Returns

Returns true if a title has been defined for this axis; else, returns false.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis
    logical :: check

    check = axis%is_title_defined()
end program
```

Definition at line 1260 of file fplot_core.f90.

6.14.2.9 procedure, public fplot_core::plot_axis::set_autoscale ( )

Sets a logical value determining if the axis should be automatically scaled to fit the data.

Syntax

```
subroutine set_autoscale(class(plot_axis) this, logical x)
```

Parameters

in,out	*this*	The plot_axis object.
in	*x*	Set to true if the axis should be automatically scaled; else, false.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis

    call axis%set_autoscale(.true.)
end program
```

Definition at line 1313 of file fplot_core.f90.

6.14.2.10 procedure, public fplot_core::plot_axis::set_is_log_scaled ( )

Sets a logical value defining if the axis should be log scaled.

Syntax

```
subroutine set_is_log_scaled(class(plot_axis) this, logical x)
```

Parameters

in,out	*this*	The plot_axis object.
in	*x*	Set to true if log scaling is applied to the axis; else, false.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis

    call axis%set_is_log_scaled(.true.)
end program
```

Definition at line 1420 of file fplot_core.f90.

6.14.2.11 procedure, public fplot_core::plot_axis::set_limits ( )

Sets the axis display limits, assuming autoscaling is not active for this axis.

Syntax

```
subroutine set_limits(class(plot_axis) this, real(real64) lower, real(real64) upper)
```

Parameters

in,out	*this*	The plot_axis object.
in	*lower*	The lower display limit.
in	*upper*	The upper display limit.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(x_axis) :: axis

    call axis%set_limits(0.0d0, 5.0d0)
end program
```

Definition at line 1368 of file fplot_core.f90.

6.14.2.12 procedure, public fplot_core::plot_axis::set_title ( )

Sets the axis' title.

Syntax

```
subroutine set_title(class(plot_axis) this, character(len = *) txt)
```

Parameters

in,out	*this*	The plot_axis object.
in	*txt*	The axis title. The number of characters must be less than or equal to PLOTDATA_MAX_NAME_LENGTH; else, the text string is truncated.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis

    call axis%set_title("X Axis")
end program
```

Definition at line 1233 of file fplot_core.f90.

6.14.2.13 procedure, public fplot_core::plot_axis::set_zero_axis ( )

Sets a value determining if the axis should be drawn through zero of opposing axes.

Syntax

```
subroutine set_zero_axis(class(plot_axis) this, logical x)
```

Parameters

in,out	*this*	The plot_axis object.
in	*x*	Set to true to draw as a zero axis; else, set to false.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    implicit none

    type(x_axis) :: axis

    call axis%get_zero_axis(.true.)
end program
```

Definition at line 1482 of file fplot_core.f90.

6.14.2.14 procedure, public fplot_core::plot_axis::set_zero_axis_line_width ( )

Sets the width of the line used to represent the zero axis line, if active.

Syntax

```
subroutine set_zero_axis_line_width(class(plot_axis) this, real(real32) x)
```

Parameters

in,out	*this*	The plot_axis object.
in	*x*	The width of the line, in pixels.

Example

Notice, this example uses an x_axis type. Any type that derives from the plot_axis type can be used.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(x_axis) :: axis

    call axis%get_zero_axis_line_width(3.0)
end program
```

Definition at line 1535 of file fplot_core.f90.

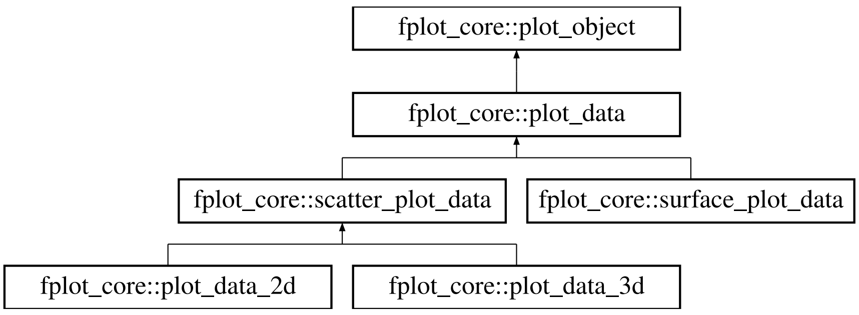
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.15 fplot_core::plot_data Type Reference

Provides a container for plot data.

Inheritance diagram for fplot_core::plot_data:



Public Member Functions

- procedure, public get_name => pd_get_name
  Gets the name to associate with this data set.
- procedure, public set_name => pd_set_name
  Sets the name to associate with this data set.
- procedure(pd_get_string_result), deferred, public get_data_string
  Gets the GNUPLOT command string containing the actual data to plot.

Private Attributes

- character(len=plotdata_max_name_length) m_name = ""

    The name of the data set.

6.15.1 Detailed Description

Provides a container for plot data.

Definition at line 1094 of file fplot_core.f90.

6.15.2 Member Function/Subroutine Documentation

6.15.2.1 procedure, public fplot_core::plot_data::get_name ( )

Gets the name to associate with this data set.

Syntax

```
character(len = :) function, allocatable get_name(class(plot_data) this)
```

Parameters

in	*this*	The plot_data object.

Returns

The name.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_data) :: pd
    character(len = :), allocatable :: name

    ! Get the name
    name = pd%get_name()
end program
```

Definition at line 1121 of file fplot_core.f90.

6.15.2.2 procedure, public fplot_core::plot_data::set_name ( )

Sets the name to associate with this data set.

Syntax

```
subroutine set_name(class(plot_data) this, character(len = *) txt)
```

Parameters

`in,out`	*this*	The plot_data object.
`in`	*txt*	The name.

Example

```
program example
    use fplot_core
    implicit none

    type(plot_data) :: pd

    ! Set the name
    call pd%set_name("Example Data Set")
end program
```

Definition at line 1143 of file fplot_core.f90.

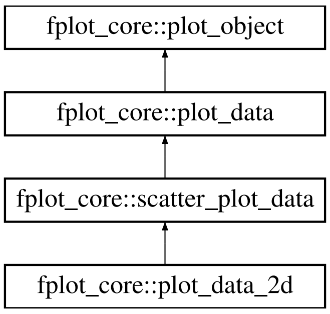
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.16 fplot_core::plot_data_2d Type Reference

Defines a two-dimensional plot data set.

Inheritance diagram for fplot_core::plot_data_2d:



Public Member Functions

- procedure, public get_axes_string => pd2d_get_axes_cmd

  Gets the GNUPLOT command string defining which axes the data is to be plotted against.
- procedure, public get_data_string => pd2d_get_data_cmd

  Gets the GNUPLOT command string containing the actual data to plot.
- procedure, public get_count => pd2d_get_data_count

  Gets the number of data points.
- procedure, public get_x => pd2d_get_x_data

  Gets the requested X data point.
- procedure, public set_x => pd2d_set_x_data

  Sets the requested X data point.
- procedure, public get_y => pd2d_get_y_data

Gets the requested Y data point.
- procedure, public set_y => pd2d_set_y_data

  Sets the requested Y data point.
- procedure, public get_draw_against_y2 => pd2d_get_draw_against_y2

  Gets a value determining if the data should be plotted against the secondary y-axis.
- procedure, public set_draw_against_y2 => pd2d_set_draw_against_y2

  Sets a value determining if the data should be plotted against the secondary y-axis.
- generic, public define_data => pd2d_set_data_1, pd2d_set_data_2

  Defines the data set.

Private Member Functions

- procedure **pd2d_set_data_1**
- procedure **pd2d_set_data_2**

Private Attributes

- real(real64), dimension(:,:), allocatable m_data

  An N-by-2 matrix containing the x and y data points.
- logical m_usey2 = .false.

  Draw against the secondary y axis?

6.16.1 Detailed Description

Defines a two-dimensional plot data set.

Definition at line 3732 of file fplot_core.f90.

6.16.2 Member Function/Subroutine Documentation

6.16.2.1 generic, public fplot_core::plot_data_2d::define_data ( )

Defines the data set.

Overload 1

Syntax

```
subroutine define_data(class(plot_data_2d) this, real(real64) x(:), real(real64) y(:), optional
          class(errors) err)
```

Parameters

`in,out`	*this*	The plot_data_2d object.
`in`	*x*	An N-element array containing the x coordinate data.
`in`	*y*	An N-element array containing the y coordinate data.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Example

The following example illustrates the use of the first overload. This form of the routine simply plots the supplied y coordinate data against the supplied x coordinate data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    ! Local Variables
    integer(int32), parameter :: npts = 1000
    real(real64), dimension(npts) :: x, y
    type(plot_2d) :: plt
    type(plot_data_2d) :: dataset
    class(plot_axis), pointer :: xaxis, yaxis
    type(legend), pointer :: leg

    ! Build a data set
    x = linspace(0.0d0, 10.0d0, npts)
    y = sin(10.0d0 * x) * sin(0.5d0 * x)

    call dataset%define_data(x, y)

    ! Set up the plot
    call plt%initialize()
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    ! Hide the legend
    leg => plt%get_legend()
    call leg%set_is_visible(.false.)

    ! Add the data to the plot
    call plt%push(dataset)

    ! Draw
    call plt%draw()
end program
```

Overload 2

Syntax

```
subroutine define_data(class(plot_data_2d) this, real(real64) y(:), optional class(errors) err)
```

Parameters

`in,out`	*this*	The plot_data_2d object.
`in`	*y*	An N-element array containing the y-coordinate data. This data will be plotted against its own index.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Example

The following example illustrates the use of the second overload. This form of the routine simply plots the data against its array index (one-based).

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    ! Local Variables
    integer(int32), parameter :: npts = 1000
    real(real64), dimension(npts) :: x, y
    type(plot_2d) :: plt
    type(plot_data_2d) :: dataset
    class(plot_axis), pointer :: xaxis, yaxis
    type(legend), pointer :: leg

    ! Build a data set
    x = linspace(0.0d0, 10.0d0, npts)
    y = sin(10.0d0 * x) * sin(0.5d0 * x)

    call dataset%define_data(y)

    ! Set up the plot
    call plt%initialize()
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    ! Hide the legend
    leg => plt%get_legend()
    call leg%set_is_visible(.false.)

    ! Add the data to the plot
    call plt%push(dataset)

    ! Draw
    call plt%draw()
end program
```

Definition at line 4118 of file fplot_core.f90.

6.16.2.2 procedure, public fplot_core::plot_data_2d::get_axes_string ( )

Gets the GNUPLOT command string defining which axes the data is to be plotted against.

Syntax

```
character(len = :) function, allocatable get_axis_string(class(plot_data_2d) this)
```

Parameters

in	*this*	The plot_data_2d object.

Returns

The command string.

Definition at line 3749 of file fplot_core.f90.

6.16.2.3 procedure, public fplot_core::plot_data_2d::get_count ( )

Gets the number of data points.

Syntax

```
pure integer(int32) get_count(class(plot_data_2d) this)
```

Parameters

in	*this*	The plot_data_2d object.

Returns

The number of data points.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    integer(int32) :: n

    ! Get the number of stored data points
    n = pd%get_count()
end program
```

Definition at line 3785 of file fplot_core.f90.

6.16.2.4 procedure, public fplot_core::plot_data_2d::get_data_string ( )

Gets the GNUPLOT command string containing the actual data to plot.

Syntax

```
character(len = :) function, allocatable get_data_string(class(plot_data_2d) this)
```

Parameters

in	*this*	The plot_data_2d object.

Returns

The command string.

Definition at line 3760 of file fplot_core.f90.

6.16.2.5 procedure, public fplot_core::plot_data_2d::get_draw_against_y2 ( )

Gets a value determining if the data should be plotted against the secondary y-axis.

Syntax

```
pure logical function get_draw_against_y2(class(plot_data_2d) this)
```

Parameters

in	*this*	The plot_data_2d object.

Returns

Returns true if the data should be plotted against the secondary y-axis; else, false to plot against the primary y-axis.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    logical :: check

    ! Determine if this data set is plotted against the secondary
    ! y axis.
    check = pd%get_draw_against_y2()
end program
```

Definition at line 3915 of file fplot_core.f90.

6.16.2.6 procedure, public fplot_core::plot_data_2d::get_x ( )

Gets the requested X data point.

Syntax

```
pure real(real64) get_x(class(plot_data_2d) this, integer(int32) index)
```

Parameters

in	*this*	The plot_data_2d object.
in	*index*	The index of the data point to retrieve.

Returns

The requested data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    real(real64) :: x

    ! Get the x data point at the 100th index
    x = pd%get_x(100)
end program
```

Definition at line 3811 of file fplot_core.f90.

6.16.2.7 procedure, public fplot_core::plot_data_2d::get_y ( )

Gets the requested Y data point.

Syntax

```
pure real(real64) get_y(class(plot_data_2d) this, integer(int32) index)
```

Parameters

in	*this*	The plot_data_2d object.
in	*index*	The index of the data point to retrieve.

Returns

The requested data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    real(real64) :: y

    ! Get the y data point at the 100th index
    y = pd%get_y(100)
end program
```

Definition at line 3862 of file fplot_core.f90.

6.16.2.8 procedure, public fplot_core::plot_data_2d::set_draw_against_y2 ( )

Sets a value determining if the data should be plotted against the secondary y-axis.

Syntax

```
subroutine set_draw_against_y2(class(plot_data_2d) this, logical x)
```

Parameters

in,out	*this*	The plot_data_2d object.
in	*x*	Set to true if the data should be plotted against the secondary y-axis; else, false to plot against the primary y-axis.

Example

This example illustrates the use of a secondary y axis.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    ! Local Variables
    integer(int32), parameter :: npts = 1000
    real(real64), dimension(npts) :: x, y1, y2
    type(plot_2d) :: plt
    type(plot_data_2d) :: ds1, ds2
    class(plot_axis), pointer :: xaxis, yaxis, y2axis

    ! Build a data set
    x = linspace(0.0d0, 10.0d0, npts)
    y1 = exp(-0.5d0 * x) * abs(sin(x))
    y2 = cos(0.5d0 * x) * sin(10.0d0 * x)

    call ds1%define_data(x, y1)
    call ds1%set_name("f(x) = exp(-x / 2) * |sin(x)|")
    call ds2%define_data(x, y2)
    call ds2%set_name("f(x) = cos(x / 2) * sin(10 x)")

    ! Make the ds2 line green and dashed
    call ds2%set_line_color(clr_green)
    call ds2%set_line_style(line_dashed)

    ! Draw ds2 against the secondary y axis
    call ds2%set_draw_against_y2(.true.)

    ! Ensure the plot knows it needs a secondary y axis
    call plt%set_use_y2_axis(.true.)

    ! Set up the plot
    call plt%initialize()
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    y2axis => plt%get_y2_axis()
    call y2axis%set_title("Secondary Y Axis")

    ! Add the data to the plot
    call plt%push(ds1)
    call plt%push(ds2)

    ! Draw
    call plt%draw()
end program
```

Definition at line 3986 of file fplot_core.f90.

6.16.2.9 procedure, public fplot_core::plot_data_2d::set_x ( )

Sets the requested X data point.

Syntax

```
subroutine set_x(class(plot_data_2d) this, integer(int32) index, real(real64) x)
```

Parameters

in,out	*this*	The plot_data_2d object.
in	*index*	The index of the data point to replace.
in	*x*	The data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd

    ! Set the x data point at the 100th index
    call pd%set_x(100, 1.25d0)
end program
```

Definition at line 3836 of file fplot_core.f90.

6.16.2.10 procedure, public fplot_core::plot_data_2d::set_y ( )

Sets the requested Y data point.

Syntax

```
subroutine set_y(class(plot_data_2d) this, integer(int32) index, real(real64) x)
```

Parameters

`in,out`	*this*	The plot_data_2d object.
`in`	*index*	The index of the data point to replace.
`in`	*x*	The data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd

    ! Set the y data point at the 100th index
    call pd%set_y(100, 1.25d0)
end program
```

Definition at line 3887 of file fplot_core.f90.

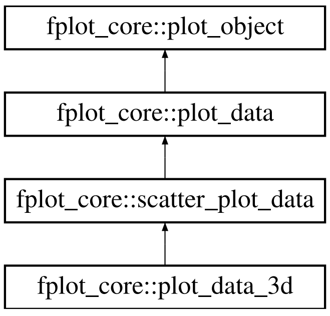
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.17 fplot_core::plot_data_3d Type Reference

Defines a three-dimensional plot data set.

Inheritance diagram for fplot_core::plot_data_3d:



Public Member Functions

- procedure, public get_count => pd3d_get_data_count

  Gets the number of data points.
- procedure, public get_x => pd3d_get_x_data

  Gets the requested X data point.
- procedure, public set_x => pd3d_set_x_data

  Sets the requested X data point.
- procedure, public get_y => pd3d_get_y_data

  Gets the requested Y data point.
- procedure, public set_y => pd3d_set_y_data

  Sets the requested Y data point.
- procedure, public get_z => pd3d_get_z_data

  Gets the requested Z data point.
- procedure, public set_z => pd3d_set_z_data

  Sets the requested Z data point.
- procedure, public get_axes_string => pd3d_get_axes_cmd

  Gets the GNUPLOT command string defining which axes the data is to be plotted against.
- procedure, public get_data_string => pd3d_get_data_cmd

  Gets the GNUPLOT command string containing the actual data to plot.
- procedure, public define_data => pd3d_set_data_1

  Defines the data set.

Private Attributes

- real(real64), dimension(:,:), allocatable m_data

  An N-by-3 matrix containing the x, y, and z data points.

6.17.1 Detailed Description

Defines a three-dimensional plot data set.

Definition at line 4191 of file fplot_core.f90.

6.17.2 Member Function/Subroutine Documentation

6.17.2.1 procedure, public fplot_core::plot_data_3d::define_data ( )

Defines the data set.

Syntax

```
subroutine define_data(class(plot_data_3d) this, real(real64) x(:), real(real64) y(:), real(real64) z(:),
        optional class(errors) err)
```

Parameters

in,out	*this*	The plot_data_2d object.
in	*x*	An N-element array containing the x coordinate data.
in	*y*	An N-element array containing the y coordinate data.

Parameters

in	*z*	An N-element array containing the z coordinate data.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available. • PLOT_ARRAY_SIZE_MISMATCH_ERROR: Occurs if `x`, `y`, and `z` are not the same size.

Example

The following example adds data to draw a helix to a 3D plot.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: n = 1000

    ! Local Variables
    real(real64), dimension(n) :: t, x, y, z
    type(plot_3d) :: plt
    type(plot_data_3d) :: d1
    class(plot_axis), pointer :: xaxis, yaxis, zaxis
    type(legend), pointer :: leg

    ! Initialize the plot object
    call plt%initialize()
    leg => plt%get_legend()
    call leg%set_is_visible(.false.)

    ! Define titles
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    zaxis => plt%get_z_axis()
    call zaxis%set_title("Z Axis")

    ! Define the data
    t = linspace(0.0d0, 10.0d0, n)
    x = cos(5.0d0 * t)
    y = sin(5.0d0 * t)
    z = 2.0d0 * t

    call d1%define_data(x, y, z)

    ! Set up the data set
    call d1%set_line_color(clr_blue)
    call d1%set_line_width(2.0)

    ! Add the data to the plot
    call plt%push(d1)

    ! Let GNUPLOT draw the plot
    call plt%draw()
end program
```

Definition at line 4471 of file fplot_core.f90.

6.17.2.2 procedure, public fplot_core::plot_data_3d::get_axes_string ( )

Gets the GNUPLOT command string defining which axes the data is to be plotted against.

Syntax

```
character(len = :) function, allocatable :: get_axes_string(class(plot_data_3d) this)
```

Parameters

in	*this*	The plot_data_3d object.

Returns

The command string.

Definition at line 4384 of file fplot_core.f90.

6.17.2.3 procedure, public fplot_core::plot_data_3d::get_count ( )

Gets the number of data points.

Syntax

```
pure integer(int32) function get_count(class(plot_data_3d) this)
```

Parameters

in	*this*	The plot_data_3d object.

Returns

The number of data points.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_3d) :: pd
    integer(int32) :: n

    ! Get the number of stored data points
    n = pd%get_count()
end program
```

Definition at line 4220 of file fplot_core.f90.

6.17.2.4 procedure, public fplot_core::plot_data_3d::get_data_string ( )

Gets the GNUPLOT command string containing the actual data to plot.

Syntax

```
character(len = :) function, allocatable :: get_data_string(class(plot_data_3d) this)
```

Parameters

in	*this*	The plot_data_3d object.

Returns

The command string.

Definition at line 4395 of file fplot_core.f90.

6.17.2.5 procedure, public fplot_core::plot_data_3d::get_x ( )

Gets the requested X data point.

Syntax

```
pure real(real64) function get_x(class(plot_data_3d), this, integer(int32) index)
```

Parameters

in	*this*	The plot_data_3d object.
in	*index*	The index of the data point to retrieve.

Returns

The requested data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_3d) :: pd
    real(real64) :: x

    ! Get the 10th value from the x-coordinate data
    x = pd%get_x(10)
end program
```

Definition at line 4246 of file fplot_core.f90.

6.17.2.6 procedure, public fplot_core::plot_data_3d::get_y ( )

Gets the requested Y data point.

Syntax

```
pure real(real64) function get_y(class(plot_data_3d) this, this, integer(int32) index)
```

Parameters

in	*this*	The plot_data_3d object.
in	*index*	The index of the data point to retrieve.

Returns

The requested data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_3d) :: pd
    real(real64) :: y

    ! Get the 10th value from the y-coordinate data
    y = pd%get_y(10)
end program
```

Definition at line 4297 of file fplot_core.f90.

6.17.2.7 procedure, public fplot_core::plot_data_3d::get_z ( )

Gets the requested Z data point.

Syntax

```
pure real(real64) function get_z(class(plot_data_3d) this, this, integer(int32) index)
```

Parameters

in	*this*	The plot_data_3d object.
in	*index*	The index of the data point to retrieve.

Returns

The requested data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_3d) :: pd
    real(real64) :: z

    ! Get the 10th value from the z-coordinate data
    z = pd%get_z(10)
end program
```

Definition at line 4348 of file fplot_core.f90.

6.17.2.8 procedure, public fplot_core::plot_data_3d::set_x ( )

Sets the requested X data point.

Syntax

```
subroutine set_x(class(plot_data_3d) this, integer(int32) index, real(real64) x)
```

Parameters

in,out	*this*	The plot_data_3d object.
in	*index*	The index of the data point to replace.
in	*x*	The data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_3d) :: pd

    ! Set the 10th value in the x-coordinate data
    call pd%set_x(10, 50.0d0)
end program
```

Definition at line 4271 of file fplot_core.f90.

6.17.2.9 procedure, public fplot_core::plot_data_3d::set_y ( )

Sets the requested Y data point.

Syntax

```
subroutine set_y(class(plot_data_3d) this, integer(int32) index, real(real64) x)
```

Parameters

in,out	*this*	The plot_data_3d object.
in	*index*	The index of the data point to replace.
in	*x*	The data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_3d) :: pd

    ! Set the 10th value in the y-coordinate data
    call pd%set_y(10, 50.0d0)
end program
```

Definition at line 4322 of file fplot_core.f90.

6.17.2.10 procedure, public fplot_core::plot_data_3d::set_z ( )

Sets the requested Z data point.

Syntax

```
subroutine set_z(class(plot_data_3d) this, integer(int32) index, real(real64) x)
```

Parameters

in,out	*this*	The plot_data_3d object.
in	*index*	The index of the data point to replace.
in	*x*	The data point.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_3d) :: pd

    ! Set the 10th value in the z-coordinate data
    call pd%set_z(10, 50.0d0)
end program
```

Definition at line 4373 of file fplot_core.f90.

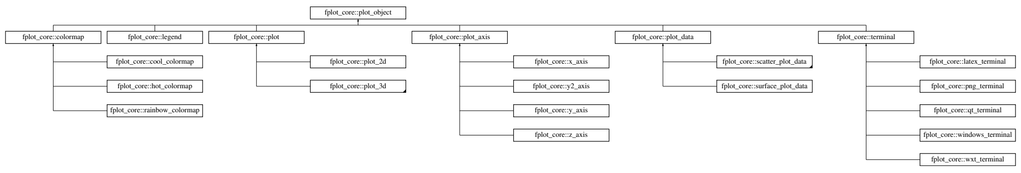
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.18 fplot_core::plot_object Type Reference

The base type for a GNUPLOT object.

Inheritance diagram for fplot_core::plot_object:



Public Member Functions

- procedure(get_string_result), deferred, public get_command_string

  Returns the appropriate GNUPLOT command string to define the plot object properties.

6.18.1 Detailed Description

The base type for a GNUPLOT object.

Definition at line 196 of file fplot_core.f90.

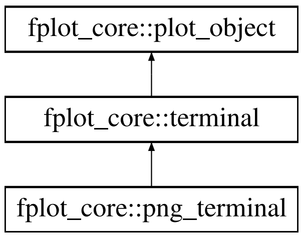
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.19 fplot_core::png_terminal Type Reference

Defines a GNUPLOT PNG terminal object.

Inheritance diagram for fplot_core::png_terminal:



Public Member Functions

- procedure, public get_filename => png_get_filename

  Gets the filename for the output PNG file.
- procedure, public set_filename => png_set_filename

  Sets the filename for the output PNG file.
- procedure, public get_id_string => png_get_term_string

  Retrieves a GNUPLOT terminal identifier string.
- procedure, public get_command_string => png_get_command_string

  Returns the appropriate GNUPLOT command string to establish appropriate parameters.

Private Attributes

- character(len=3) m_id = "png"

  The terminal ID string.
- character(len=gnuplot_max_path_length) m_fname = "default.png"

  The filename of the PNG file to write.

6.19.1 Detailed Description

Defines a GNUPLOT PNG terminal object.

Example

The following example draws a simple plot, and illustrates the use of a png_terminal to draw directly to a PNG file.

```
program example
    use iso_fortran_env
    use fplot_core
    implicit none

    ! Local Variables & Parameters
    integer(int32), parameter :: npts = 1000
    real(real64), dimension(npts) :: x, y1, y2
    type(plot_2d) :: plt
    class(terminal), pointer :: term
    type(plot_data_2d) :: d1, d2
    class(plot_axis), pointer :: xaxis, yaxis
    type(legend), pointer :: leg

    ! Build a data set to plot
    x = linspace(0.0d0, 10.0d0, npts)
    y1 = sin(x) * cos(x)
    y2 = sqrt(x) * sin(x)

    call d1%define_data(x, y1)
    call d2%define_data(x, y2)

    ! Set up the plot
    call plt%initialize(gnuplot_terminal_png) ! Save to file directly
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    ! Put the legend in the upper left corner of the plot
    leg => plt%get_legend()
    call leg%set_horizontal_position(legend_left)
    call leg%set_vertical_position(legend_top)

    ! Set up line color and style properties to better distinguish each data set
    call d1%set_name("Data Set 1")
    call d1%set_line_color(clr_blue)

    call d2%set_name("Data Set 2")
    call d2%set_line_color(clr_green)

    ! Add the data to the plot
    call plt%push(d1)
    call plt%push(d2)

    ! Define the file to which the plot should be saved
    term => plt%get_terminal()
    select type (term)
    class is (png_terminal)
        call term%set_filename("example_plot.png")
    end select

    ! Draw the plot
    call plt%draw()
end program
```

Definition at line 890 of file fplot_core.f90.

6.19.2 Member Function/Subroutine Documentation

6.19.2.1 procedure, public fplot_core::png_terminal::get_command_string ( )

Returns the appropriate GNUPLOT command string to establish appropriate parameters.

Syntax

```
character(len = :) function, allocatable get_command_string(class(png_terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The GNUPLOT command string.

Definition at line 962 of file fplot_core.f90.

6.19.2.2 procedure, public fplot_core::png_terminal::get_filename ( )

Gets the filename for the output PNG file.

Syntax

```
character(len = :) function, allocatable get_filename(class(png_terminal) this)
```

Parameters

in	*this*	The png_terminal object.

Returns

The filename, including the file extension (.png).

Example

```
program example
    use fplot_core
    implicit none

    type(png_terminal) :: term
    character(len = :), allocatable :: fname

    ! Get the filename
    fname = term%get_filename()
end program
```

Definition at line 919 of file fplot_core.f90.

6.19.2.3 procedure, public fplot_core::png_terminal::get_id_string ( )

Retrieves a GNUPLOT terminal identifier string.

Syntax

```
character(len = :) function, allocatable get_id_string(class(png_terminal) this)
```

Parameters

in	*this*	The png_terminal object.

Returns

The string.

Definition at line 951 of file fplot_core.f90.

6.19.2.4 procedure, public fplot_core::png_terminal::set_filename ( )

Sets the filename for the output PNG file.

Syntax

```
subroutine set_filename(class(png_terminal) this, character(len = *) txt)
```

Parameters

in,out	*this*	The png_terminal object.
in	*txt*	The filename, including the file extension (.png).

Example

```
program example
    use fplot_core
    implicit none

    type(png_terminal) :: term

    ! Set the filename
    call term%set_filename("Example PNG File.png")
end program
```

Definition at line 941 of file fplot_core.f90.

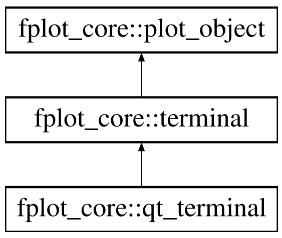
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.20 fplot_core::qt_terminal Type Reference

Defines a GNUPLOT QT terminal object.

Inheritance diagram for fplot_core::qt_terminal:



Public Member Functions

- procedure, public get_id_string => qt_get_term_string

  Retrieves a GNUPLOT terminal identifier string.

Private Attributes

- character(len=2) m_id = "qt"

  The terminal ID string.

6.20.1 Detailed Description

Defines a GNUPLOT QT terminal object.

Definition at line 767 of file fplot_core.f90.

6.20.2 Member Function/Subroutine Documentation

6.20.2.1 procedure, public fplot_core::qt_terminal::get_id_string ( )

Retrieves a GNUPLOT terminal identifier string.

Syntax

```
character(len = :) function, allocatable get_id_string(class(qt_terminal) this)
```

Parameters

in	*this*	The qt_terminal object.

Returns

The string.

Definition at line 781 of file fplot_core.f90.

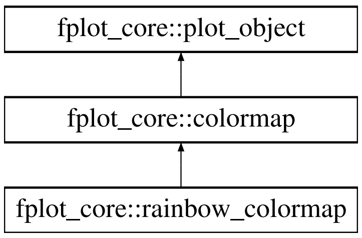
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.21 fplot_core::rainbow_colormap Type Reference

Defines a rainbow colormap.

Inheritance diagram for fplot_core::rainbow_colormap:



Public Member Functions

- procedure, public get_color_string => rcm_get_clr

  Gets the GNUPLOT string defining the color distribution.

6.21.1 Detailed Description

Defines a rainbow colormap.

Example

The following example illustrates a surface plot using a rainbow colormap.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    type(rainbow_colormap) :: map ! Using a rainbow colormap
    class(plot_axis), pointer :: xaxis, yaxis, zaxis

    ! Define the data
    xdata = linspace(xmin, xmax, n)
    ydata = linspace(ymin, ymax, m)
    xy = meshgrid(xdata, ydata)
    x => xy(:,:,1)
    y => xy(:,:,2)

    ! Define the function to plot
    z = sin(sqrt(x**2 + y**2))

    ! Create the plot
    call plt%initialize()
    call plt%set_colormap(map)

    ! Define titles
    call plt%set_title("Surface Example Plot 1")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    zaxis => plt%get_z_axis()
    call zaxis%set_title("Z Axis")
```

```
        ! Define the data set
        call d1%define_data(x, y, z)
        call d1%set_name("sin(sqrt(x**2 + y**2))")
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 2892 of file fplot_core.f90.

6.21.2 Member Function/Subroutine Documentation

6.21.2.1 procedure, public fplot_core::rainbow_colormap::get_color_string ( )

Gets the GNUPLOT string defining the color distribution.

Syntax

```
    character(len = :) function, allocatable get_color_string(class(rainbow_colormap) this)
```

Parameters

in	*this*	The rainbow_colormap object.

Returns

The command string.

Definition at line 2903 of file fplot_core.f90.

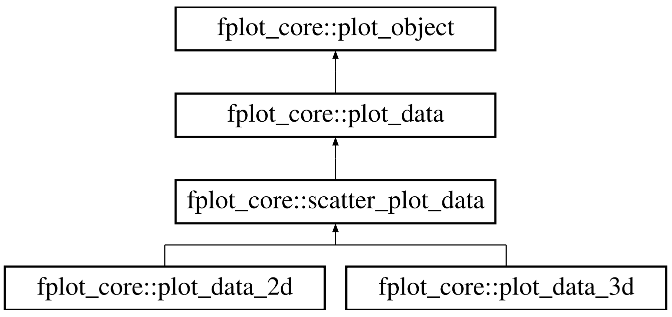
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.22 fplot_core::scatter_plot_data Type Reference

A plot_data object for describing scatter plot data sets.

Inheritance diagram for fplot_core::scatter_plot_data:



Public Member Functions

- procedure, public get_command_string => spd_get_cmd

  Gets the GNUPLOT command string to represent this scatter_plot_data object.
- procedure, public get_line_width => spd_get_line_width

  Gets the width of the line, in pixels.
- procedure, public set_line_width => spd_set_line_width

  Sets the width of the line, in pixels.
- procedure, public get_line_style => spd_get_line_style

  Gets the line style.
- procedure, public set_line_style => spd_set_line_style

  Sets the line style.
- procedure, public get_line_color => spd_get_line_color

  Gets the line color.
- procedure, public set_line_color => spd_set_line_color

  Sets the line color.
- procedure, public get_draw_line => spd_get_draw_line

  Gets a value determining if a line should be drawn.
- procedure, public set_draw_line => spd_set_draw_line

  Sets a value determining if a line should be drawn.
- procedure, public get_draw_markers => spd_get_draw_markers

  Gets a value determining if data point markers should be drawn.
- procedure, public set_draw_markers => spd_set_draw_markers

  Sets a value determining if data point markers should be drawn.
- procedure, public get_marker_style => spd_get_marker_style

  Gets the marker style.
- procedure, public set_marker_style => spd_set_marker_style

  Sets the marker style.
- procedure, public get_marker_scaling => spd_get_marker_scaling

  Gets the marker scaling.
- procedure, public set_marker_scaling => spd_set_marker_scaling

  Sets the marker scaling.
- procedure, public get_marker_frequency => spd_get_marker_frequency

  Gets the marker frequency.
- procedure, public set_marker_frequency => spd_set_marker_frequency

  Sets the marker frequency.
- procedure, public get_use_auto_color => spd_get_use_auto_colors

  Gets a value determining if GNUPLOT should automatically choose line colors.
- procedure, public set_use_auto_color => spd_set_use_auto_colors

  Sets a value determining if GNUPLOT should automatically choose line colors.
- procedure(spd_get_int_value), deferred, public get_count

  Gets the number of data points.
- procedure(spd_get_value), deferred, public get_x

  Gets the requested X data point.
- procedure(spd_set_value), deferred, public set_x

  Sets the requested X data point.
- procedure(spd_get_value), deferred, public get_y

  Gets the requested Y data point.
- procedure(spd_set_value), deferred, public set_y

  Sets the requested X data point.
- procedure(spd_get_string_result), deferred, public get_axes_string

  Gets the GNUPLOT command string defining which axes the data is to be plotted against.

Private Attributes

- logical m_drawline = .true.

  Draw the line?
- logical m_drawmarkers = .false.

  Draw the markers?
- integer(int32) m_markerfrequency = 1

  Marker frequency.
- type(color) m_linecolor = CLR_BLUE

  Line color.
- real(real32) m_linewidth = 1.0

  Line width.
- integer(int32) m_linestyle = LINE_SOLID

  Line style.
- integer(int32) m_markertype = MARKER_X

  Marker type.
- real(real32) m_markersize = 1.0

  Marker size multiplier.
- logical m_useautocolor = .false.

  Let GNUPLOT choose colors automatically.

6.22.1 Detailed Description

A plot_data object for describing scatter plot data sets.

Definition at line 3095 of file fplot_core.f90.

6.22.2 Member Function/Subroutine Documentation

6.22.2.1 procedure, public fplot_core::scatter_plot_data::get_command_string ( )

Gets the GNUPLOT command string to represent this scatter_plot_data object.

Syntax

```
character(len = :) function, allocatable get_command_string(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

The command string.

Definition at line 3126 of file fplot_core.f90.

6.22.2.2 procedure, public fplot_core::scatter_plot_data::get_draw_line ( )

Gets a value determining if a line should be drawn.

Syntax

```
pure logical function get_draw_line(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

Returns true if the line should be drawn; else, false.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    logical :: check

    ! Check to see if a line should be drawn to connect data points
    check = pd%get_draw_line()
end program
```

Definition at line 3323 of file fplot_core.f90.

6.22.2.3 procedure, public fplot_core::scatter_plot_data::get_draw_markers ( )

Gets a value determining if data point markers should be drawn.

Syntax

```
pure logical function get_draw_markers(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

Returns true if the markers should be drawn; else, false.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    logical :: check

    ! Check to see if markers should be drawn at data points
    check = pd%get_draw_markers()
end program
```

Definition at line 3377 of file fplot_core.f90.

6.22.2.4 procedure, public fplot_core::scatter_plot_data::get_line_color ( )

Gets the line color.

Syntax

```
pure type(color) function get_line_color(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

The color.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    type(color) :: clr

    ! Get the line color
    clr = pd%get_line_color()
end program
```

Definition at line 3270 of file fplot_core.f90.

6.22.2.5 procedure, public fplot_core::scatter_plot_data::get_line_style ( )

Gets the line style.

Syntax

```
pure integer(int32) function get_line_style(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

The line style. The line style must be one of the following:

- LINE_DASHED
- LINE_DASH_DOTTED
- LINE_DASH_DOT_DOT
- LINE_DOTTED
- LINE_SOLID

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    integer(int32) :: style

    ! Get the line style
    style = pd%get_line_style()
end program
```

Definition at line 3211 of file fplot_core.f90.

6.22.2.6 procedure, public fplot_core::scatter_plot_data::get_line_width ( )

Gets the width of the line, in pixels.

Syntax

```
pure real(real32) function get_line_width(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

The line width.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
```

```
        implicit none

        type(plot_data_2d) :: pd
        real(real32) :: width

        ! Get the line width
        width = pd%get_line_width()
    end program
```

Definition at line 3153 of file fplot_core.f90.

6.22.2.7 procedure, public fplot_core::scatter_plot_data::get_marker_frequency ( )

Gets the marker frequency.

Syntax

```
    pure integer(int32) function get_marker_frequency(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

The marker frequency.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
    program example
        use fplot_core
        use iso_fortran_env
        implicit none

        type(plot_data_2d) :: pd
        integer(int32) :: freq

        ! Get the data point marker frequency
        freq = pd%get_marker_frequency()
    end program
```

Definition at line 3565 of file fplot_core.f90.

6.22.2.8 procedure, public fplot_core::scatter_plot_data::get_marker_scaling ( )

Gets the marker scaling.

Syntax

```
    pure real(real32) function get_marker_scaling(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

The scaling factor.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    real(real32) :: scaling

    ! Get the data point marker scaling factor
    scaling = pd%get_marker_scaling()
end program
```

Definition at line 3511 of file fplot_core.f90.

6.22.2.9 procedure, public fplot_core::scatter_plot_data::get_marker_style ( )

Gets the marker style.

Syntax

```
pure integer(int32) function get_marker_style(class(scatter_plot_data) this)
```

Parameters

in	*this*	The scatter_plot_data object.

Returns

The marker type. The marker type must be one of the following:

- MARKER_ASTERISK
- MARKER_EMPTY_CIRCLE
- MARKER_EMPTY_NABLA
- MARKER_EMPTY_RHOMBUS
- MARKER_EMPTY_SQUARE
- MARKER_EMPTY_TRIANGLE
- MARKER_FILLED_CIRCLE
- MARKER_FILLED_NABLA
- MARKER_FILLED_RHOMBUS
- MARKER_FILLED_SQUARE
- MARKER_FILLED_TRIANGLE
- MARKER_PLUS
- MARKER_X

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    integer(int32) :: marker

    ! Get the data point marker style
    marker = pd%get_marker_style()
end program
```

Definition at line 3444 of file fplot_core.f90.

6.22.2.10 procedure, public fplot_core::scatter_plot_data::get_use_auto_color ( )

Gets a value determining if GNUPLOT should automatically choose line colors.

Syntax

```
pure logical function get_use_auto_color(class(scatter_plot_data) this)
```

Parameters

`in`	*this*	The scatter_plot_data object.

Returns

Returns true if GNUPLOT should choose colors; else, false.

Definition at line 3603 of file fplot_core.f90.

6.22.2.11 procedure, public fplot_core::scatter_plot_data::set_draw_line ( )

Sets a value determining if a line should be drawn.

Syntax

```
subroutine set_draw_line(class(scatter_plot_data) this, logical x)
```

Parameters

`in,out`	*this*	The scatter_plot_data object.
`in`	*x*	Set to true if the line should be drawn; else, false.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none
```

```
        type(plot_data_2d) :: pd

        ! Force a line to be drawn between data points
        call pd%set_draw_line(.true.)
    end program
```

Definition at line 3349 of file fplot_core.f90.

6.22.2.12 procedure, public fplot_core::scatter_plot_data::set_draw_markers ( )

Sets a value determining if data point markers should be drawn.

Syntax

```
    subroutine set_draw_markers(class(scatter_plot_data) this, logical x)
```

Parameters

`in,out`	*this*	The scatter_plot_data object.
`in`	*x*	Set to true if the markers should be drawn; else, false.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
    program example
        use fplot_core
        use iso_fortran_env
        implicit none

        type(plot_data_2d) :: pd

        ! Force markers to be drawn at data points
        call pd%set_draw_markers(.true.)
    end program
```

Definition at line 3404 of file fplot_core.f90.

6.22.2.13 procedure, public fplot_core::scatter_plot_data::set_line_color ( )

Sets the line color.

Syntax

```
    subroutine set_line_color(class(scatter_plot_data) this, type(color) x)
```

Parameters

`in,out`	*this*	The scatter_plot_data object.
`in`	*x*	The color.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd

    ! Set the line color to red
    call pd%set_line_color(clr_red)
end program
```

Definition at line 3296 of file fplot_core.f90.

6.22.2.14 procedure, public fplot_core::scatter_plot_data::set_line_style ( )

Sets the line style.

Syntax

```
subroutine set_line_style(class(scatter_plot_data) this, integer(int32) x)
```

Parameters

`in,out`	*this*	The scatter_plot_data object.
`in`	*x*	The line style. The line style must be one of the following: • LINE_DASHED • LINE_DASH_DOTTED • LINE_DASH_DOT_DOT • LINE_DOTTED • LINE_SOLID

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd

    ! Set the line style
    call pd%get_line_style(line_dashed)
end program
```

Definition at line 3243 of file fplot_core.f90.

6.22.2.15 procedure, public fplot_core::scatter_plot_data::set_line_width ( )

Sets the width of the line, in pixels.

Syntax

```
subroutine set_line_width(class(scatter_plot_data) this, real(real32) x)
```

Parameters

in,out	*this*	The scatter_plot_data object.
in	*x*	The line width.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd

    ! Set the line width
    call pd%set_line_width(2.0)
end program
```

Definition at line 3179 of file fplot_core.f90.

6.22.2.16 procedure, public fplot_core::scatter_plot_data::set_marker_frequency ( )

Sets the marker frequency.

Syntax

```
subroutine set_marker_frequency(class(scatter_plot_data) this, integer(int32) x)
```

Parameters

in,out	*this*	The scatter_plot_data object.
in	*x*	The marker frequency.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd
    real(real32) :: scaling

    ! Set a data point marker every second data point
    call pd%set_marker_frequency(2)
end program
```

Definition at line 3592 of file fplot_core.f90.

6.22.2.17 procedure, public fplot_core::scatter_plot_data::set_marker_scaling ( )

Sets the marker scaling.

Syntax

```
subroutine set_marker_scaling(class(scatter_plot_data) this, real(real32) x)
```

Parameters

in,out	*this*	The scatter_plot_data object.
in	*x*	The scaling factor.

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd

    ! Set the data point marker scaling factor such that the marker
    ! is scaled by a factor of 2
    call pd%set_marker_scaling(2.0)
end program
```

Definition at line 3538 of file fplot_core.f90.

6.22.2.18 procedure, public fplot_core::scatter_plot_data::set_marker_style ( )

Sets the marker style.

Syntax

```
subroutine set_marker_style(class(scatter_plot_data) this, integer(int32) x)
```

Parameters

in,out	*this*	The scatter_plot_data object.
in	*x*	The marker type. The marker type must be one of the following: • MARKER_ASTERISK • MARKER_EMPTY_CIRCLE • MARKER_EMPTY_NABLA • MARKER_EMPTY_RHOMBUS • MARKER_EMPTY_SQUARE • MARKER_EMPTY_TRIANGLE • MARKER_FILLED_CIRCLE • MARKER_FILLED_NABLA • MARKER_FILLED_RHOMBUS • MARKER_FILLED_SQUARE • MARKER_FILLED_TRIANGLE • MARKER_PLUS • MARKER_X

Example

This example makes use of the plot_data_2d type; however, this example is valid for any type that derives from scatter_plot_data.

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(plot_data_2d) :: pd

    ! Set the data point marker style to a plus (+) sign
    call pd%set_marker_style(marker_plus)
end program
```

Definition at line 3484 of file fplot_core.f90.

6.22.2.19 procedure, public fplot_core::scatter_plot_data::set_use_auto_color ( )

Sets a value determining if GNUPLOT should automatically choose line colors.

Syntax

```
subroutine set_use_auto_color(class(scatter_plot_data) this, logical x)
```

Parameters

`in,out`	*this*	The scatter_plot_data object.
`in`	*x*	Set to true if GNUPLOT should choose colors; else, false.

Definition at line 3614 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.23 fplot_core::spd_get_int_value Interface Reference

Retrieves an integer value from a scatter_plot_data object.

Private Member Functions

- pure integer(int32) function **spd_get_int_value** (this)

6.23.1 Detailed Description

Retrieves an integer value from a scatter_plot_data object.

Parameters

`in`	*this*	The scatter_plot_data object.

Returns

The requested value.

Definition at line 6753 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.24 fplot_core::spd_get_string_result Interface Reference

Retrieves a string from a scatter_plot_data object.

Private Member Functions

- character(len=:) function, allocatable **spd_get_string_result** (this)

6.24.1 Detailed Description

Retrieves a string from a scatter_plot_data object.

Parameters

in	*this*	The scatter_plot_data object.

Returns

The string.

Definition at line 6764 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.25 fplot_core::spd_get_value Interface Reference

Retrieves a numeric value from a scatter_plot_data object.

Private Member Functions

- pure real(real64) function **spd_get_value** (this, index)

6.25.1 Detailed Description

Retrieves a numeric value from a scatter_plot_data object.

Parameters

`in`	*this*	The scatter_plot_data object.
`in`	*index*	The index of the value to retrieve.

Returns

The requested value.

Definition at line 6728 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.26 fplot_core::spd_set_value Interface Reference

Sets a numeric value into a scatter_plot_data object.

Private Member Functions

- subroutine **spd_set_value** (this, index, x)

6.26.1 Detailed Description

Sets a numeric value into a scatter_plot_data object.

Parameters

`in,out`	*this*	The scatter_plot_data object.
`in`	*index*	The index of the value to retrieve.
`in`	*x*	The value.

Definition at line 6741 of file fplot_core.f90.

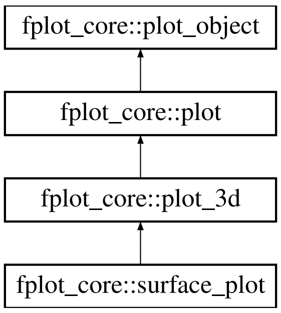
The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.27 fplot_core::surface_plot Type Reference

A plot object defining a 3D surface plot.

Inheritance diagram for fplot_core::surface_plot:



Public Member Functions

- procedure, public initialize => surf_init

  Initializes the surface_plot object.
- procedure, public get_show_hidden => surf_get_show_hidden

  Gets a value indicating if hidden lines should be shown.
- procedure, public set_show_hidden => surf_set_show_hidden

  Sets a value indicating if hidden lines should be shown.
- procedure, public get_command_string => surf_get_cmd

  Gets the GNUPLOT command string to represent this plot_3d object.
- procedure, public get_colormap => surf_get_colormap

  Gets a pointer to the colormap object.
- procedure, public set_colormap => surf_set_colormap

  Sets the colormap object.
- procedure, public get_allow_smoothing => surf_get_smooth

  Gets a value determining if the plotted surfaces should be smoothed.
- procedure, public set_allow_smoothing => surf_set_smooth

  Sets a value determining if the plotted surfaces should be smoothed.
- procedure, public get_show_contours => surf_get_show_contours

  Gets a value determining if a contour plot should be drawn in conjunction with the surface plot.
- procedure, public set_show_contours => surf_set_show_contours

  Sets a value determining if a contour plot should be drawn in conjunction with the surface plot.
- procedure, public get_show_colorbar => surf_get_show_colorbar

  Gets a value determining if the colorbar should be shown.
- procedure, public set_show_colorbar => surf_set_show_colorbar

  Sets a value determining if the colorbar should be shown.
- procedure, public get_use_lighting => surf_get_use_lighting

  Gets a value indicating if lighting, beyond the ambient light source, is to be used.
- procedure, public set_use_lighting => surf_set_use_lighting

  Sets a value indicating if lighting, beyond the ambient light source, is to be used.
- procedure, public get_light_intensity => surf_get_light_intensity

  Gets the ratio of the strength of the light source relative to the ambient light.
- procedure, public set_light_intensity => surf_set_light_intensity

  Sets the ratio of the strength of the light source relative to the ambient light.
- procedure, public get_specular_intensity => surf_get_specular_intensity

  Gets the ratio of the strength of the specular light source relative to the ambient light.
- procedure, public set_specular_intensity => surf_set_specular_intensity

  Sets the ratio of the strength of the specular light source relative to the ambient light.

Private Member Functions

- final surf_clean_up

  Cleans up resources held by the surface_plot object.

Private Attributes

- logical m_showhidden = .false.

  Show hidden lines.
- class(colormap), pointer m_colormap

  The colormap.
- logical m_smooth = .true.

  Smooth the surface?
- logical m_contour = .false.

  Show a contour plot as well as the surface plot?
- logical m_showcolorbar = .true.

  Show the colorbar?
- logical m_uselighting = .false.

  Use lighting?
- real(real32) m_lightintensity = 0.5

  Lighting intensity (0 - 1) - default is 0.5.
- real(real32) m_specular = 0.5

  Specular highlight intensity (0 - 1)

6.27.1 Detailed Description

A plot object defining a 3D surface plot.

Example

The following example illustrates a surface plot using a rainbow colormap.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    class(plot_axis), pointer :: xaxis, yaxis, zaxis

    ! Define the data
    xdata = linspace(xmin, xmax, n)
    ydata = linspace(ymin, ymax, m)
    xy = meshgrid(xdata, ydata)
    x => xy(:,:,1)
    y => xy(:,:,2)

    ! Define the function to plot
    z = sin(sqrt(x**2 + y**2))
```

```
        ! Create the plot
        call plt%initialize()

        ! Define titles
        call plt%set_title("Surface Example Plot 1")

        xaxis => plt%get_x_axis()
        call xaxis%set_title("X Axis")

        yaxis => plt%get_y_axis()
        call yaxis%set_title("Y Axis")

        zaxis => plt%get_z_axis()
        call zaxis%set_title("Z Axis")

        ! Define the data set
        call d1%define_data(x, y, z)
        call d1%set_name("sin(sqrt(x**2 + y**2))")
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 5875 of file fplot_core.f90.

6.27.2 Member Function/Subroutine Documentation

6.27.2.1 procedure, public fplot_core::surface_plot::get_allow_smoothing ( )

Gets a value determining if the plotted surfaces should be smoothed.

Syntax

```
pure logical function get_allow_smoothing(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

Returns true if the surface should be smoothed; else, false.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    logical :: check

    ! Check to see if the surfaces should be smoothed by interpolation.
    check = plt%get_allow_smoothing()
end program
```

Definition at line 6115 of file fplot_core.f90.

6.27.2.2 procedure, public fplot_core::surface_plot::get_colormap ( )

Gets a pointer to the colormap object.

Syntax

```
class(colormap) function, pointer get_colormap(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

A pointer to the colormap object. If no colormap is defined, a null pointer is returned.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    class(colormap), pointer :: map

    ! Get a pointer to the current colormap
    map => plt%get_colormap()
end program
```

Definition at line 6058 of file fplot_core.f90.

6.27.2.3 procedure, public fplot_core::surface_plot::get_command_string ( )

Gets the GNUPLOT command string to represent this plot_3d object.

Syntax

```
character(len = :) function, allocatable get_command_string(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

The command string.

Definition at line 6033 of file fplot_core.f90.

6.27.2.4 procedure, public fplot_core::surface_plot::get_light_intensity ( )

Gets the ratio of the strength of the light source relative to the ambient light.

Syntax

```
pure real(real32) function get_light_intensity(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

The light intensity ratio.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot) :: plt
    real(real32) :: val

    ! Get the lighting intensity
    val = plt%get_light_intensity()
end program
```

Definition at line 6423 of file fplot_core.f90.

6.27.2.5 procedure, public fplot_core::surface_plot::get_show_colorbar ( )

Gets a value determining if the colorbar should be shown.

Syntax

```
pure logical function get_show_colorbar(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

Returns true if the colorbar should be drawn; else, false.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    logical :: check

    ! Check to see if the colorbar is shown
    check = plt%get_show_colorbar()
end program
```

Definition at line 6269 of file fplot_core.f90.

6.27.2.6 procedure, public fplot_core::surface_plot::get_show_contours ( )

Gets a value determining if a contour plot should be drawn in conjunction with the surface plot.

Syntax

```
pure logical function get_show_contours(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

Returns true if the contour plot should be drawn; else, false to only draw the surface.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    logical :: check

    ! Check to see if contour lines are to be drawn
    check = plt%get_show_countours()
end program
```

Definition at line 6165 of file fplot_core.f90.

6.27.2.7 procedure, public fplot_core::surface_plot::get_show_hidden ( )

Gets a value indicating if hidden lines should be shown.

Syntax

```
pure logical function get_show_hidden(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

Returns true if hidden lines should be shown; else, false.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    logical :: check

    ! Check to see if hidden lines are to be shown
    check = plt%get_show_hidden()
end program
```

Definition at line 5942 of file fplot_core.f90.

6.27.2.8 procedure, public fplot_core::surface_plot::get_specular_intensity ( )

Gets the ratio of the strength of the specular light source relative to the ambient light.

Syntax

```
pure real(real32) function get_specular_intensity(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

The specular light intensity ratio.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot) :: plt
    real(real32) :: val

    ! Get the lighting intensity
    val = plt%get_specular_intensity()
end program
```

Definition at line 6464 of file fplot_core.f90.

6.27.2.9 procedure, public fplot_core::surface_plot::get_use_lighting ( )

Gets a value indicating if lighting, beyond the ambient light source, is to be used.

Syntax

```
pure logical function get_use_lighting(class(surface_plot) this)
```

Parameters

in	*this*	The surface_plot object.

Returns

True if lighting should be used; else, false.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    logical :: check

    ! Determine if lighting is to be used
    check = plt%get_use_lighting()
end program
```

Definition at line 6318 of file fplot_core.f90.

6.27.2.10 procedure, public fplot_core::surface_plot::initialize ( )

Initializes the surface_plot object.

Syntax

```
subroutine initialize(class(surface_plot) this, optional integer(int32) term, optional class(errors) err)
```

Parameters

in	*this*	The surface_plot object.
in	*term*	An optional input that is used to define the terminal. The default terminal is a WXT terminal. The acceptable inputs are: • GNUPLOT_TERMINAL_PNG • GNUPLOT_TERMINAL_QT • GNUPLOT_TERMINAL_WIN32 • GNUPLOT_TERMINAL_WXT
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Definition at line 5918 of file fplot_core.f90.

6.27.2.11 procedure, public fplot_core::surface_plot::set_allow_smoothing ( )

Sets a value determining if the plotted surfaces should be smoothed.

Syntax

```
subroutine set_allow_smoothing(class(surface_plot) this, logical x)
```

Parameters

in,out	*this*	The surface_plot object.
in	*x*	Set to true if the surface should be smoothed; else, false.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt

    ! Turn off smoothing (the default is on)
    call plt%set_allow_smoothing(.false.)
end program
```

Definition at line 6139 of file fplot_core.f90.

6.27.2.12 procedure, public fplot_core::surface_plot::set_colormap ( )

Sets the colormap object.

Syntax

```
subroutine set_colormap(class(surface_plot) this, class(colormap) x, optional class(errors) err)
```

Parameters

`in,out`	*this*	The surface_plot object.
`in`	*x*	The colormap object. Notice, a copy of this object is stored, and the surface_plot object then manages the lifetime of the copy.
`out`	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows. • PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    type(rainbow_colormap) :: map

    ! Set the colormap to a rainbow colormap
    call plt%set_colormap(map)
end program
```

Definition at line 6090 of file fplot_core.f90.

6.27.2.13 procedure, public fplot_core::surface_plot::set_light_intensity ( )

Sets the ratio of the strength of the light source relative to the ambient light.

Syntax

```
subroutine set_light_intensity(class(surface_plot) this, real(real32) x)
```

Parameters

`in,out`	*this*	The surface_plot object.
`in`	*x*	The light intensity ratio. The value must exist in the set [0, 1]; else, it will be clipped to lie within the range.

Example

See set_use_lighting for example useage.

Definition at line 6438 of file fplot_core.f90.

6.27.2.14 procedure, public fplot_core::surface_plot::set_show_colorbar ( )

Sets a value determining if the colorbar should be shown.

Syntax

```
subroutine set_show_colorbar(class(surface_plot) this, logical x)
```

Parameters

`in,out`	*this*	The surface_plot object.
`in`	*x*	Set to true if the colorbar should be drawn; else, false.

Example

```
program example
    use fplot_core
    implicit none

    type(surface_plot) :: plt
    logical :: check

    ! Hide the colorbar
    call plt%set_show_colorbar(.false.)
end program
```

Definition at line 6293 of file fplot_core.f90.

6.27.2.15 procedure, public fplot_core::surface_plot::set_show_contours ( )

Sets a value determining if a contour plot should be drawn in conjunction with the surface plot.

Syntax

```
subroutine set_show_contours(class(surface_plot) this, logical x)
```

Parameters

`in,out`	*this*	The surface_plot object.
`in`	*x*	Set to true if the contour plot should be drawn; else, false to only draw the surface.

Example

The following example illustrates the use of a contour and surface plot together. Additionally, the z axis is allowed to shift away from the X-Y plane in order to better show the counter plot.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    type(rainbow_colormap) :: map
    class(plot_axis), pointer :: xaxis, yaxis, zaxis

    ! Define the data
    xdata = linspace(xmin, xmax, n)
```

```
        ydata = linspace(ymin, ymax, m)
        xy = meshgrid(xdata, ydata)
        x => xy(:,:,1)
        y => xy(:,:,2)

        ! Define the function to plot
        z = sin(sqrt(x**2 + y**2))

        ! Create the plot
        call plt%initialize()
        call plt%set_colormap(map)
        call plt%set_show_contours(.true.)
        call plt%set_z_intersect_xy(.false.)

        ! Define titles
        call plt%set_title("Example Plot")

        xaxis => plt%get_x_axis()
        call xaxis%set_title("X Axis")

        yaxis => plt%get_y_axis()
        call yaxis%set_title("Y Axis")

        zaxis => plt%get_z_axis()
        call zaxis%set_title("Z Axis")

        ! Define the data set
        call d1%define_data(x, y, z)
        call d1%set_name("sin(sqrt(x**2 + y**2))")
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 6245 of file fplot_core.f90.

6.27.2.16 procedure, public fplot_core::surface_plot::set_show_hidden ( )

Sets a value indicating if hidden lines should be shown.

Syntax

```
subroutine set_show_hidden(class(surface_plot) this, logical x)
```

Parameters

in,out	*this*	The surface_plot object.
in	*x*	Set to true if hidden lines should be shown; else, false.

Example

The following example illustrates the use of hidden lines. The default wireframe behavior is to hide hidden lines.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
```

```
        real(real64), dimension(m, n, 2), target :: xy
        real(real64), dimension(m, n) :: z
        type(surface_plot) :: plt
        type(surface_plot_data) :: d1
        class(plot_axis), pointer :: xaxis, yaxis, zaxis

        ! Define the data
        xdata = linspace(xmin, xmax, n)
        ydata = linspace(ymin, ymax, m)
        xy = meshgrid(xdata, ydata)
        x => xy(:,:,1)
        y => xy(:,:,2)

        ! Define the function to plot
        z = sin(sqrt(x**2 + y**2))

        ! Create the plot
        call plt%initialize()
        call plt%set_show_hidden(.true.)
        call d1%set_use_wireframe(.true.)

        ! Set up lighting
        call plt%set_use_lighting(.true.)
        call plt%set_light_intensity(0.7)
        call plt%set_specular_intensity(0.7)

        ! Define titles
        call plt%set_title("Example Plot")

        xaxis => plt%get_x_axis()
        call xaxis%set_title("X Axis")

        yaxis => plt%get_y_axis()
        call yaxis%set_title("Y Axis")

        zaxis => plt%get_z_axis()
        call zaxis%set_title("Z Axis")

        ! Define the data set
        call d1%define_data(x, y, z)
        call d1%set_name("sin(sqrt(x**2 + y**2))")
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 6022 of file fplot_core.f90.

6.27.2.17 procedure, public fplot_core::surface_plot::set_specular_intensity ( )

Sets the ratio of the strength of the specular light source relative to the ambient light.

Syntax

```
    subroutine set_specular_intensity(class(surface_plot) this, real(real32) x)
```

Parameters

in,out	*this*	The surface_plot object.
in	*x*	The specular light intensity ratio. The value must exist in the set [0, 1]; else, it will be clipped to lie within the range.

Example

See set_use_lighting for example useage.

Definition at line 6480 of file fplot_core.f90.

6.27.2.18 procedure, public fplot_core::surface_plot::set_use_lighting ( )

Sets a value indicating if lighting, beyond the ambient light source, is to be used.

Syntax

```
subroutine set_use_lighting(class(surface_plot) this, logical x)
```

Parameters

in,out	*this*	The surface_plot object.
in	*x*	True if lighting should be used; else, false.

Example

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    type(rainbow_colormap) :: map
    class(plot_axis), pointer :: xaxis, yaxis, zaxis

    ! Define the data
    xdata = linspace(xmin, xmax, n)
    ydata = linspace(ymin, ymax, m)
    xy = meshgrid(xdata, ydata)
    x => xy(:,:,1)
    y => xy(:,:,2)

    ! Define the function to plot
    z = sin(sqrt(x**2 + y**2))

    ! Create the plot
    call plt%initialize()
    call plt%set_colormap(map)

    ! Set up lighting
    call plt%set_use_lighting(.true.)
    call plt%set_light_intensity(0.7)
    call plt%set_specular_intensity(0.7)

    ! Define titles
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    zaxis => plt%get_z_axis()
    call zaxis%set_title("Z Axis")

    ! Define the data set
    call d1%define_data(x, y, z)
    call d1%set_name("sin(sqrt(x**2 + y**2))")
    call plt%push(d1)
```

```
        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 6397 of file fplot_core.f90.

6.27.2.19 final fplot_core::surface_plot::surf_clean_up ( ) [final], [private]

Cleans up resources held by the surface_plot object.

Parameters

in,out	*this*	The surface_plot object.

Definition at line 5897 of file fplot_core.f90.

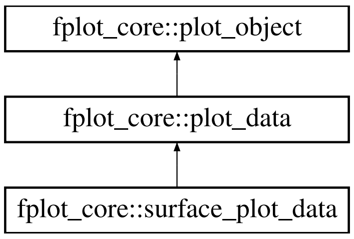
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.28 fplot_core::surface_plot_data Type Reference

Provides a three-dimensional surface plot data set.

Inheritance diagram for fplot_core::surface_plot_data:



Public Member Functions

- procedure, public get_size => surfd_get_size

  Gets the size of the stored data set.
- procedure, public get_x => surfd_get_x

  Gets the requested X data point.
- procedure, public set_x => surfd_set_x

  Sets the requested X data point.
- procedure, public get_y => surfd_get_y

  Gets the requested Y data point.
- procedure, public set_y => surfd_set_y

  Sets the requested Y data point.
- procedure, public get_z => surfd_get_z

  Gets the requested Z data point.
- procedure, public set_z => surfd_set_z

Sets the requested Z data point.

- procedure, public get_use_wireframe => surfd_get_wireframe

  Gets a value determining if a wireframe mesh should be displayed.
- procedure, public set_use_wireframe => surfd_set_wireframe

  Sets a value determining if a wireframe mesh should be displayed.
- procedure, public get_command_string => surfd_get_cmd

  Gets the GNUPLOT command string to represent this surface_plot_data object.
- procedure, public get_data_string => surfd_get_data_cmd

  Gets the GNUPLOT command string containing the actual data to plot.
- procedure, public define_data => surfd_set_data_1

  Defines the data set.

Private Attributes

- real(real64), dimension(:,:), allocatable m_x

  Stores the x-coordinate data.
- real(real64), dimension(:,:), allocatable m_y

  Stores the y-coordinate data.
- real(real64), dimension(:,:), allocatable m_z

  Stores the z-coordinate data.
- logical m_wireframe = .false.

  Set to true to display a wireframe of the surface; else, just a smooth surface will be drawn.

6.28.1 Detailed Description

Provides a three-dimensional surface plot data set.

Definition at line 4538 of file fplot_core.f90.

6.28.2 Member Function/Subroutine Documentation

6.28.2.1 procedure, public fplot_core::surface_plot_data::define_data ( )

Defines the data set.

Syntax

```
subroutine define_data(class(surface_plot_data) this, real(real64) x(:,:), real(real64) y(:,:),
        real(real64) z(:,:))
```

Parameters

in,out	*this*	The surface_plot_data object.
in	*x*	An M-by-N matrix containing the x-coordinate data.
in	*y*	An M-by-N matrix containing the y-coordinate data.
in	*z*	An M-by-N matrix containing the z-coordinate data.
out	*err*	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		• PLOT_OUT_OF_MEMORY_ERROR: Occurs if insufficient memory is available. • PLOT_ARRAY_SIZE_MISMATCH_ERROR: Occurs if `x`, `y`, and `z` are not the same size.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50

    ! Local Variables
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), pointer, dimension(:,:) :: x, y
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    class(plot_axis), pointer :: xaxis, yaxis, zaxis
    type(rainbow_colormap) :: map

    ! Define the data
    xy = meshgrid(linspace(-5.0d0, 5.0d0, n), linspace(-5.0d0, 5.0d0, m))
    x => xy(:,:,1)
    y => xy(:,:,2)

    ! Initialize the plot
    call plt%initialize()
    call plt%set_colormap(map)

    ! Set the orientation of the plot
    call plt%set_elevation(20.0d0)
    call plt%set_azimuth(30.0d0)

    ! Define titles
    call plt%set_title("Example Plot")

    xaxis => plt%get_x_axis()
    call xaxis%set_title("X Axis")

    yaxis => plt%get_y_axis()
    call yaxis%set_title("Y Axis")

    zaxis => plt%get_z_axis()
    call zaxis%set_title("Z Axis")

    ! Define the function to plot
    z = sqrt(x**2 + y**2) * sin(x**2 + y**2)
    call d1%define_data(x, y, z)
    call plt%push(d1)

    ! Draw the plot
    call plt%draw()
end program
```

Definition at line 4941 of file fplot_core.f90.

6.28.2.2 procedure, public fplot_core::surface_plot_data::get_command_string ( )

Gets the GNUPLOT command string to represent this surface_plot_data object.

Syntax

```
character(len = :) function, allocatable get_command_string(class(surface_plot_data) this)
```

Parameters

in	*this*	The surface_plot_data object.

Returns

The command string.

Definition at line 4854 of file fplot_core.f90.

6.28.2.3 procedure, public fplot_core::surface_plot_data::get_data_string ( )

Gets the GNUPLOT command string containing the actual data to plot.

Syntax

```
character(len = :) function, allocatable get_data_string(class(surface_plot_data) this)
```

Parameters

in	*this*	The surface_plot_data object.

Returns

The GNUPLOT command string.

Definition at line 4865 of file fplot_core.f90.

6.28.2.4 procedure, public fplot_core::surface_plot_data::get_size ( )

Gets the size of the stored data set.

Syntax

```
pure integer(int32) function get_size(class(surface_plot_data) this, integer(int32) dim)
```

Parameters

in	*this*	The suface_plot_data object.
in	*dim*	The dimension of interest. Notice, data is stored as a 2D matrix (i.e. only 1 and 2 are valid inputs).

Returns

The size of the requested dimension.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot_data) :: pd
    integer(int32) :: nrows, ncols

    ! Get the number of rows in the data matrices
    nrows = pd%get_size(1)

    ! Get the number of columns in the data matrices
    ncols = pd%get_size(2)
end program
```

Definition at line 4579 of file fplot_core.f90.

6.28.2.5 procedure, public fplot_core::surface_plot_data::get_use_wireframe ( )

Gets a value determining if a wireframe mesh should be displayed.

Syntax

```
pure logical function get_wireframe(class(surface_plot_data) this)
```

Parameters

in	*this*	The surface_plot_data object.

Returns

Returns true if a wireframe mesh should be displayed; else, false to display a solid surface.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot_data) :: pd
    logical :: check

    ! Check to see if the data set is to be plotted in wireframe
    check = pd%get_use_wireframe()
end program
```

Definition at line 4768 of file fplot_core.f90.

6.28.2.6 procedure, public fplot_core::surface_plot_data::get_x ( )

Gets the requested X data point.

Syntax

```
pure real(real64) function get_x(class(surface_plot_data) this, integer(int32) i, integer(int32) j)
```

Parameters

in	*this*	The surface_plot_data object.
in	*i*	The row index.
in	*j*	The column index.

Returns

The value.

Example

```
program example
    use fplot_core
```

```
        use iso_fortran_env
        implicit none

        type(surface_plot_data) :: pd
        real(real64) :: val

        ! Get a value from the 10th row and 15th column of the X data
        val = pd%get_x(10, 15)
    end program
```

Definition at line 4606 of file fplot_core.f90.

6.28.2.7 procedure, public fplot_core::surface_plot_data::get_y ( )

Gets the requested Y data point.

Syntax

```
    pure real(real64) function get_y(class(surface_plot_data) this, integer(int32) i, integer(int32) j)
```

Parameters

in	*this*	The surface_plot_data object.
in	*i*	The row index.
in	*j*	The column index.

Returns

The value.

Example

```
    program example
        use fplot_core
        use iso_fortran_env
        implicit none

        type(surface_plot_data) :: pd
        real(real64) :: val

        ! Get a value from the 10th row and 15th column of the Y data
        val = pd%get_y(10, 15)
    end program
```

Definition at line 4660 of file fplot_core.f90.

6.28.2.8 procedure, public fplot_core::surface_plot_data::get_z ( )

Gets the requested Z data point.

Syntax

```
    pure real(real64) function get_z(class(surface_plot_data) this, integer(int32) i, integer(int32) j)
```

Parameters

in	*this*	The surface_plot_data object.
in	*i*	The row index.
in	*j*	The column index.

Returns

The value.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot_data) :: pd
    real(real64) :: val

    ! Get a value from the 10th row and 15th column of the Z data
    val = pd%get_z(10, 15)
end program
```

Definition at line 4714 of file fplot_core.f90.

6.28.2.9 procedure, public fplot_core::surface_plot_data::set_use_wireframe ( )

Sets a value determining if a wireframe mesh should be displayed.

Syntax

```
subroutine set_wireframe(class(surface_plot_data) this, logical x)
```

Parameters

in,out	*this*	The surface_plot_data object.
in	*x*	Set to true if a wireframe mesh should be displayed; else, false to display a solid surface.

Example

This example builds a wireframe surface plot.

```
program example
    use, intrinsic :: iso_fortran_env
    use fplot_core
    implicit none

    ! Parameters
    integer(int32), parameter :: m = 50
    integer(int32), parameter :: n = 50
    real(real64), parameter :: xmax = 5.0d0
    real(real64), parameter :: xmin = -5.0d0
    real(real64), parameter :: ymax = 5.0d0
    real(real64), parameter :: ymin = -5.0d0

    ! Local Variables
    real(real64), dimension(n) :: xdata
    real(real64), dimension(m) :: ydata
    real(real64), dimension(:,:), pointer :: x, y
    real(real64), dimension(m, n, 2), target :: xy
    real(real64), dimension(m, n) :: z
    type(surface_plot) :: plt
    type(surface_plot_data) :: d1
    class(plot_axis), pointer :: xaxis, yaxis, zaxis

    ! Define the data
    xdata = linspace(xmin, xmax, n)
    ydata = linspace(ymin, ymax, m)
    xy = meshgrid(xdata, ydata)
    x => xy(:,:,1)
    y => xy(:,:,2)

    ! Define the function to plot
    z = sin(sqrt(x**2 + y**2))
```

```
        ! Create the plot
        call plt%initialize()
        call d1%set_use_wireframe(.true.)

        ! Define titles
        call plt%set_title("Example Plot")

        xaxis => plt%get_x_axis()
        call xaxis%set_title("X Axis")

        yaxis => plt%get_y_axis()
        call yaxis%set_title("Y Axis")

        zaxis => plt%get_z_axis()
        call zaxis%set_title("Z Axis")

        ! Define the data set
        call d1%define_data(x, y, z)
        call d1%set_name("sin(sqrt(x**2 + y**2))")
        call plt%push(d1)

        ! Let GNUPLOT draw the plot
        call plt%draw()
    end program
```

Definition at line 4843 of file fplot_core.f90.

6.28.2.10 procedure, public fplot_core::surface_plot_data::set_x ( )

Sets the requested X data point.

Syntax

```
subroutine set_x(class(surface_plot_data) this, integer(int32) i, integer(int32) j, real(real64) x)
```

Parameters

in,out	*this*	The surface_plot_data object.
in	*i*	The row index.
in	*j*	The column index.
in	*x*	The value.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot_data) :: pd
    real(real64) :: val

    ! Set a value into the 10th row and 15th column of the X data
    call pd%set_x(10, 15, 5.0d0)
end program
```

Definition at line 4633 of file fplot_core.f90.

6.28.2.11 procedure, public fplot_core::surface_plot_data::set_y ( )

Sets the requested Y data point.

Syntax

```
subroutine set_y(class(surface_plot_data) this, integer(int32) i, integer(int32) j, real(real64) x)
```

Parameters

in,out	*this*	The surface_plot_data object.
in	*i*	The row index.
in	*j*	The column index.
in	*x*	The value.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot_data) :: pd
    real(real64) :: val

    ! Set a value into the 10th row and 15th column of the Y data
    call pd%set_y(10, 15, 5.0d0)
end program
```

Definition at line 4687 of file fplot_core.f90.

6.28.2.12 procedure, public fplot_core::surface_plot_data::set_z ( )

Sets the requested Z data point.

Syntax

```
subroutine set_z(class(surface_plot_data) this, integer(int32) i, integer(int32) j, real(real64) x)
```

Parameters

in,out	*this*	The surface_plot_data object.
in	*i*	The row index.
in	*j*	The column index.
in	*x*	The value.

Example

```
program example
    use fplot_core
    use iso_fortran_env
    implicit none

    type(surface_plot_data) :: pd
    real(real64) :: val

    ! Set a value into the 10th row and 15th column of the Z data
    call pd%set_z(10, 15, 5.0d0)
end program
```

Definition at line 4741 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.29 fplot_core::term_get_string_result Interface Reference

Retrieves a string from a terminal.

Private Member Functions

- character(len=:) function, allocatable **term_get_string_result** (this)

6.29.1 Detailed Description

Retrieves a string from a terminal.

Parameters

`in`	*this*	The terminal object.

Returns

The string.

Definition at line 6697 of file fplot_core.f90.

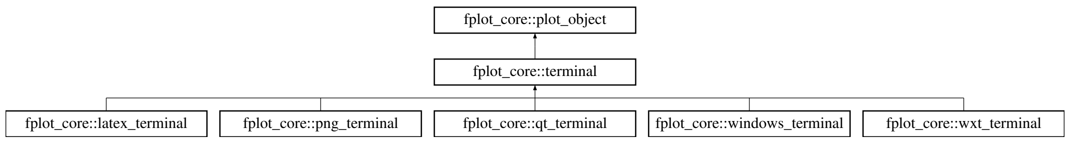
The documentation for this interface was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.30 fplot_core::terminal Type Reference

Defines a GNUPLOT terminal object.

Inheritance diagram for fplot_core::terminal:



Public Member Functions

- procedure, public get_window_width => term_get_window_width

  Gets the width of the plot window.
- procedure, public set_window_width => term_set_window_width

  Sets the width of the plot window.
- procedure, public get_window_height => term_get_window_height

  Gets the height of the plot window.
- procedure, public set_window_height => term_set_window_height

Sets the height of the plot window.
- procedure, public get_command_string => term_get_command_string

  Returns the appropriate GNUPLOT command string to establish appropriate parameters.
- procedure, public get_plot_window_number =>term_get_plot_window_number

  Gets the targeted plot window number.
- procedure, public set_plot_window_number =>term_set_plot_window_number

  Sets the targeted plot window number.
- procedure, public get_title => term_get_title

  Gets the plot window's title.
- procedure, public set_title => term_set_title

  Sets the plot window's title.
- procedure, public get_font_name => term_get_font_name

  Gets the name of the font used for text displayed by the graph.
- procedure, public set_font_name => term_set_font_name

  Sets the name of the font used for text displayed by the graph.
- procedure, public get_font_size => term_get_font_size

  Gets the size of the font used by the graph.
- procedure, public set_font_size => term_set_font_size

  Sets the size of the font used by the graph.
- procedure(term_get_string_result), deferred, public get_id_string

  Gets the GNUPLOT terminal identification string.

Private Attributes

- integer(int32) m_windowheight = GNUPLOT_DEFAULT_WINDOW_HEIGHT

  The window height, in pixels.
- integer(int32) m_windowwidth = GNUPLOT_DEFAULT_WINDOW_WIDTH

  The window width, in pixels.
- integer(int32) m_termid = 0

  The plot window number.
- character(len=gnuplot_max_label_length) m_title = ""

  The plot window title.
- logical m_hastitle = .false.

  Determines if a plot title is defined.
- character(len=gnuplot_max_label_length) m_fontname = GNUPLOT_DEFAULT_FONTNAME

  The font used by the graph.
- integer(int32) m_fontsize = GNUPLOT_DEFAULT_FONT_SIZE

  The size of the font used by the graph.

6.30.1 Detailed Description

Defines a GNUPLOT terminal object.

Definition at line 345 of file fplot_core.f90.

6.30.2 Member Function/Subroutine Documentation

6.30.2.1 procedure, public fplot_core::terminal::get_command_string ( )

Returns the appropriate GNUPLOT command string to establish appropriate parameters.

Syntax

```
character(len = :) function, allocatable get_command_string(class(terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The GNUPLOT command string.

Definition at line 481 of file fplot_core.f90.

6.30.2.2 procedure, public fplot_core::terminal::get_font_name ( )

Gets the name of the font used for text displayed by the graph.

Syntax

```
character(len = :) function, allocatable get_font_name(class(terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The font name.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term
    character(len = :), allocatable :: font

    ! Get the name of the font.
    font = term%get_font_name()
end program
```

Definition at line 581 of file fplot_core.f90.

6.30.2.3 procedure, public fplot_core::terminal::get_font_size ( )

Gets the size of the font used by the graph.

Syntax

```
pure integer(int32) function get_font_size(class(terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The font size, in points.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term
    integer(int32) :: sz

    ! Get the font size.
    sz = term%get_font_size()
end program
```

Definition at line 634 of file fplot_core.f90.

6.30.2.4 procedure, public fplot_core::terminal::get_plot_window_number ( )

Gets the targeted plot window number.

Syntax

```
pure integer(int32) function get_plot_window_number(class(terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The plot window number.

Definition at line 491 of file fplot_core.f90.

6.30.2.5 procedure, public fplot_core::terminal::get_title ( )

Gets the plot window's title.

Syntax

```
character(len = :) function, allocatable get_title(class(terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The title.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term
    character(len = :), allocatable :: title

    ! Get the plot window title.
    title = term%get_title()
end program
```

Definition at line 529 of file fplot_core.f90.

6.30.2.6 procedure, public fplot_core::terminal::get_window_height ( )

Gets the height of the plot window.

Syntax

```
pure integer(int32) function get_window_height(class(terminal) this)
```

Parameters

in	*this*	The terminal object.

Returns

The height of the plot window.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term
    integer(int32) :: height

    ! Get the height of the plot window
    height = term%get_window_height()
end program
```

Definition at line 442 of file fplot_core.f90.

6.30.2.7 procedure, public fplot_core::terminal::get_window_width ( )

Gets the width of the plot window.

Syntax

```
pure integer(int32) function get_window_width(class(terminal) this)
```

Parameters

`in`	*this*	The terminal object.

Returns

The width of the plot window.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term
    integer(int32) :: width

    ! Get the width of the plot window
    width = term%get_window_width()
end program
```

Definition at line 388 of file fplot_core.f90.

6.30.2.8 procedure, public fplot_core::terminal::set_font_name ( )

Sets the name of the font used for text displayed by the graph.

Syntax

```
subroutine set_font_name(class(terminal) this, character(len = *) name)
```

Parameters

`in,out`	*this*	The terminal object.
`in`	*name*	The name of the font. If no name is supplied, the name is reset back to its default setting.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term

    ! Get the name of the font.
    call term%set_font_name("Arial")
end program
```

Definition at line 608 of file fplot_core.f90.

6.30.2.9 procedure, public fplot_core::terminal::set_font_size ( )

Sets the size of the font used by the graph.

Syntax

```
subroutine set_font_size(class(terminal) this, integer(int32) sz)
```

Parameters

`in,out`	*this*	The terminal object.
`in`	*sz*	The font size, in points. If a value of zero is provided, the font size is reset to its default value; or, if a negative value is provided, the absolute value of the supplied value is utilized.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term

    ! Set the size of the font.
    call term%set_font_size(12)
end program
```

Definition at line 661 of file fplot_core.f90.

6.30.2.10 procedure, public fplot_core::terminal::set_plot_window_number ( )

Sets the targeted plot window number.

Syntax

```
subroutine set_plot_window_number(class(terminal) this, integer(int32) x)
```

Parameters

`in,out`	*this*	The terminal object.
`in`	*x*	The plot window number.

Definition at line 502 of file fplot_core.f90.

6.30.2.11 procedure, public fplot_core::terminal::set_title ( )

Sets the plot window's title.

Syntax

```
subroutine set_title(class(terminal) this, character(len = *) txt)
```

Parameters

`in,out`	*this*	The terminal object.
`in`	*txt*	The title.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term

    ! Set the plot window title.
    call term%set_title("New Window Title")
end program
```

Definition at line 554 of file fplot_core.f90.

6.30.2.12 procedure, public fplot_core::terminal::set_window_height ( )

Sets the height of the plot window.

Syntax

```
subroutine set_window_height(class(terminal) this, integer(int32) x)
```

Parameters

in,out	*this*	The terminal object.
in	*x*	The height of the plot window. If a value of zero is provided, the window height is reset to its default value; or, if a negative value is provided, the absolute value of the supplied value is utilized.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term

    ! Set the height of the plot window to 400 pixels.
    call term%set_window_height(400)
end program
```

Definition at line 470 of file fplot_core.f90.

6.30.2.13 procedure, public fplot_core::terminal::set_window_width ( )

Sets the width of the plot window.

Syntax

```
subroutine set_window_width(class(terminal) this, integer(int32) x)
```

Parameters

in,out	*this*	The terminal object.
in	*x*	The width of the plot window. If a value of zero is provided, the window width is reset to its default value; or, if a negative value is provided, the absolute value of the supplied value is utilized.

Example

Notice, this example uses a wxt_terminal. Any type that derives from the terminal type can be used.

```
program example
    use fplot_core
    implicit none

    type(wxt_terminal) :: term

    ! Set the width of the plot window to 400 pixels.
    call term%set_window_width(400)
end program
```

Definition at line 416 of file fplot_core.f90.

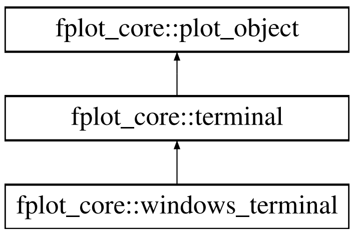
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.31 fplot_core::windows_terminal Type Reference

Defines a GNUPLOT Win32 terminal object.

Inheritance diagram for fplot_core::windows_terminal:



Public Member Functions

- procedure, public get_id_string => wt_get_term_string

    Retrieves a GNUPLOT terminal identifier string.

Private Attributes

- character(len=3) m_id = "win"

    The terminal ID string.

6.31.1 Detailed Description

Defines a GNUPLOT Win32 terminal object.

Definition at line 738 of file fplot_core.f90.

6.31.2 Member Function/Subroutine Documentation

6.31.2.1 procedure, public fplot_core::windows_terminal::get_id_string ( )

Retrieves a GNUPLOT terminal identifier string.

Syntax

```
character(len = :) function, allocatable get_id_string(class(windows_terminal) this)
```

Parameters

in	*this*	The windows_terminal object.

Returns

The string.

Definition at line 752 of file fplot_core.f90.

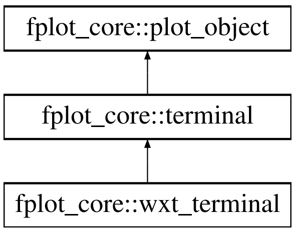
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.32 fplot_core::wxt_terminal Type Reference

Defines a GNUPLOT WXT terminal object.

Inheritance diagram for fplot_core::wxt_terminal:



Public Member Functions

- procedure, public get_id_string => wxt_get_term_string
  Retrieves a GNUPLOT terminal identifier string.

Private Attributes

- character(len=3) m_id = "wxt"
  The terminal ID string.

6.32.1 Detailed Description

Defines a GNUPLOT WXT terminal object.

Definition at line 796 of file fplot_core.f90.

6.32.2 Member Function/Subroutine Documentation

6.32.2.1 procedure, public fplot_core::wxt_terminal::get_id_string ( )

Retrieves a GNUPLOT terminal identifier string.

Syntax

```
character(len = :) function, allocatable get_id_string(class(wxt_terminal) this)
```

Parameters

in	*this*	The wxt_terminal object.

Returns

The string.

Definition at line 810 of file fplot_core.f90.

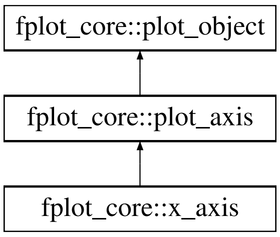
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.33 fplot_core::x_axis Type Reference

An x-axis object.

Inheritance diagram for fplot_core::x_axis:



Public Member Functions

- procedure, public get_id_string => xa_get_id

  Gets the axis identification string.

Private Attributes

- character m_id = "x"

  The ID character.

6.33.1 Detailed Description

An x-axis object.

Syntax

```
character(len = :) function, allocatable get_id_string(class(x_axis) this)
```

Parameters

in	*this*	The x_axis object.

Returns

The string.

Definition at line 6594 of file fplot_core.f90.

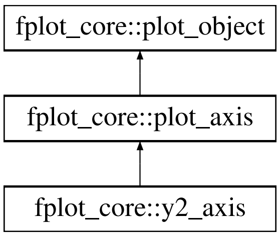
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.34 fplot_core::y2_axis Type Reference

A secondary y-axis object.

Inheritance diagram for fplot_core::y2_axis:



Public Member Functions

- procedure, public get_id_string => y2a_get_id

  Gets the axis identification string.

Private Attributes

- character(len=2) m_id = "y2"

  The ID character.

6.34.1 Detailed Description

A secondary y-axis object.

Syntax

```
character(len = :) function, allocatable get_id_string(class(y2_axis) this)
```

Parameters

in	*this*	The y2_axis object.

Returns

The string.

Definition at line 6630 of file fplot_core.f90.

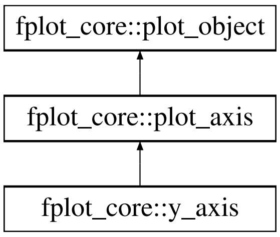
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.35 fplot_core::y_axis Type Reference

A y-axis object.

Inheritance diagram for fplot_core::y_axis:



Public Member Functions

- procedure, public get_id_string => ya_get_id

    Gets the axis identification string.

Private Attributes

- character m_id = "y"

    The ID character.

6.35.1 Detailed Description

A y-axis object.

Syntax

```
character(len = :) function, allocatable get_id_string(class(y_axis) this)
```

Parameters

in	*this*	The y_axis object.

Returns

The string.

Definition at line 6612 of file fplot_core.f90.

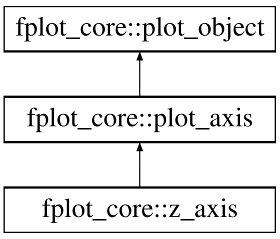
The documentation for this type was generated from the following file:

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

6.36 fplot_core::z_axis Type Reference

A z-axis object.

Inheritance diagram for fplot_core::z_axis:



Public Member Functions

- procedure, public get_id_string => za_get_id
  Gets the axis identification string.

Private Attributes

- character m_id = "z"
  The ID character.

6.36.1 Detailed Description

A z-axis object.

Syntax

```
character(len = :) function, allocatable get_id_string(class(z_axis) this)
```

Parameters

`in`	*this*	The z_axis object.

Returns

The string.

Definition at line 6648 of file fplot_core.f90.

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

- /home/jason/Documents/Code/fplot/src/fplot_core.f90

Index