Damian Kaczmarek, Maciej Kamiński

RexIO Terminal Control Library 1.0

Library reference manual

for revision number 271

April 2, 2009
www.rexio.org

rexio

1611107

Generated by Doxygen 1.5.6

Thu Apr 2 15:31:35 2009

Contents

1 Namespace Index

1.1 Namespace List

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

Scr (**Namespace of lower half of the library**)	**6**
Scr::Bg (**Background colors. WITHOUT style**)	**12**
Scr::Control (**Namespace containing iomanipulator-like items**)	**13**
Scr::Fg (**Foreground colors and styles**)	**14**
Scr::TI	**15**
Scr::Tk::Detail (**Selection form widget**)	**16**

2 Class Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

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

5 Namespace Documentation

5.1 Scr Namespace Reference

Namespace of lower half of the library.

Classes

- struct Vector

  vector container

- struct Size

  size container

- struct Position

  position container.

- class Exception

  base class for exceptions thrown by library objects.

- class GlyphWidth
- class Key

  Class represents key (or key combination) pressed on client terminal.

- class DisplayStyle

  complete set of display properties for single character

- class Screen

  Class representing basic output operation is defined as ABC (abstract base).

- class Connection

  Class representing basic input and output operations.

- class AutoList

  container combining advantages of list and hash map, allowing

- class BufferedInput

  Intermediate between Scr::__ScreenConnection and std::istream.

- class __ScreenConnection

  internal class which is base for all connection-specific implementations of screen (multiple-inheritance case)

- class RScreen

  template class representing full implementation of Scr::Screen and Scr::__ScreenConnection

- class Dictionary

  replacement of std::map<std::string,T> - optimized for string key random access using dictionary-tree data structure.

- class GenericScreen

  Most basic implementation of whole Scr::Screen.

- class LocalScreen

  connection on localhost, using cin/cout

- class RemoteScreen

  TELNET connection.

- class ScreenBase

  Implements features common to subscreen and generic screen.

- class ScreenCharacter

  character to be displayed with all it's properties

- class ScreenRow

  single row of ScreenBuffer object (which may contain more rows)

- class ScreenBuffer

  buffer of characters, supporting colours and unicode.

- class SubScreen
- class Terminal

  base class containing data fields typical to any terminal output type

- class TerminfoEnabledScreen

  class representing terminal controlled according to terminfo database

- class VT100Compatible

  terminal compatible w/ DEC VT-100

Namespaces

- namespace Bg

  Background colors. WITHOUT style.

- namespace Control

namespace containing iomanipulator-like items

- namespace Fg

  Foreground colors and styles.

- namespace TI

Typedefs

- typedef unsigned long Uint

  Machine specific unsigned integer. Type of at least 32 bits.

- typedef long Sint

  Machine specific signed integer. Type of at least 32 bits.

Enumerations

- enum

Functions

- bool operator!= (const Scr::Position &p1, const Scr::Position &p2)

  Standard comparison operator.

- bool operator== (const Scr::Position &p1, const Scr::Position &p2)

  Standard comparison operator.

- unsigned long width (wchar_t c)
- wchar_t DecodeUTF8 (const char **pstr) throw (Screen::InvalidUTF8)
- void EncodeUTF8 (std::ostream &o, Uint c) throw ()
- Uint CharLengthUTF8 (const char *s) throw (Screen::InvalidUTF8)
- Uint StringLengthUTF8 (const char *s) throw (Screen::InvalidUTF8)

Variables

- const Uint UintMax = -1

  Maximal value of Uint type.

- const Uint SintMax = UintMax/2

  Maximal value of Sint type.

- const Uint SintMin = -SintMax-1

  Minimal value of Sint type.

5.1.1 Detailed Description

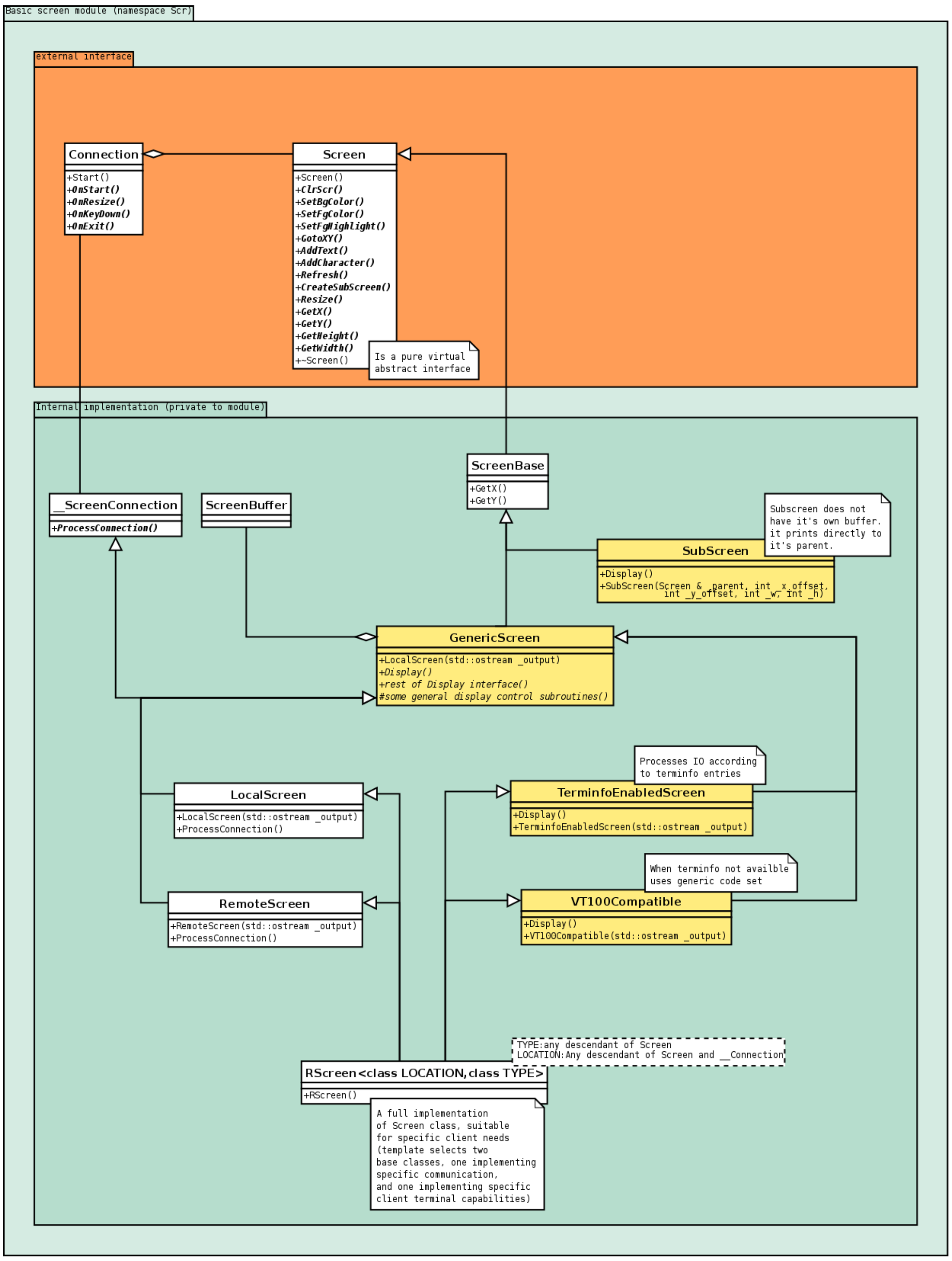
Namespace of lower half of the library.

This namespace contains classes and other utilities connected with general purpose text screen manipulation and input processing. It implements platform independent Screen class, and Connection class representing basic framework for console application development.

Note:

Scr::Tk is upper part of the library, and is recommended for all higher level UI manipulation, but Scr::Connection can be used alone

Following figure is simplified class relationship diagram for Scr::Screen and Scr::Connection connected items, focusing on internal layout of implementation of screen.


Basic screen module (namespace Scr)
external interface
Connection
+Start()
+OnStart()
+OnResize()
+OnKeyDown()
+OnExit()
Screen
+Screen()
+ClrScr()
+SetBgColor()
+SetFgColor()
+SetFgHighlight()
+GotoXY()
+AddText()
+AddCharacter()
+Refresh()
+CreateSubScreen()
+Resize()
+GetX()
+GetY()
+GetHeight()
+GetWidth()
+~Screen()
Is a pure virtual abstract interface
Internal implementation (private to module)
ScreenBase
+GetX()
+GetY()
__ScreenConnection
+ProcessConnection()
ScreenBuffer
SubScreen
+Display()
+SubScreen(Screen & _parent, int _x_offset, int _y_offset, int _w, int _h)
Subscreen does not have it's own buffer. it prints directly to it's parent.
GenericScreen
+LocalScreen(std::ostream _output)
+Display()
+rest of Display interface()
#some general display control subroutines()
LocalScreen
+LocalScreen(std::ostream _output)
+ProcessConnection()
RemoteScreen
+RemoteScreen(std::ostream _output)
+ProcessConnection()
Processes IO according to terminfo entries
TerminfoEnabledScreen
+Display()
+TerminfoEnabledScreen(std::ostream _output)
When terminfo not availble uses generic code set
VT100Compatible
+Display()
+VT100Compatible(std::ostream _output)
TYPE:any descendant of Screen
LOCATION:Any descendant of Screen and __Connection
RScreen<class LOCATION,class TYPE>
+RScreen()
A full implementation of Screen class, suitable for specific client needs (template selects two base classes, one implementing specific communication, and one implementing specific client terminal capabilities)

5.1.2 Enumeration Type Documentation

5.1.2.1 anonymous enum

Logging levels acceptable for RexIOLog macro

5.1.3 Function Documentation

5.1.3.1 Uint Scr::CharLengthUTF8 (const char ∗ *s*) throw (Screen::InvalidUTF8)

Parameters:

s UTF-8 string

Compute number of bytes in UTF-8 encoding of the FIRST character of UTF-8 string.

Note:

function assumes, that string is correct. No validation or range checking is performed

5.1.3.2 wchar_t Scr::DecodeUTF8 (const char ∗∗ *pstr*) throw (Screen::InvalidUTF8)

Parameters:

pstr pointer to NULL-terminated c-style string.

Returns:

RAW UNICODE value of utf8 encoded first character of string supplied.

if length of u8 code is greater than 1 byte, pstr is moved by this length-1 forward.

Exceptions:

Scr::Screen::InvalidFirstByte is thrown when ∗∗pstr (or pstr[0][0]) does not match 1-byte, 2-byte, 3-byte nor 4-byte UTF-8 encoding pattern for first byte.

Scr::Screen::OverlongUTF8Encoding is thrown when numeric value of result would fit in smaller number of bytes with correct UTF-8.

Scr::Screen::InvalidTrailingByte is thrown if second or maybe third or fourth byte does not match template (exactly (c[x]&0xC0)!=0x80)

Note:

if compiled without -DDO_VALIDATE_UTF_8_OUTPUT, none of theese exceptions is thrown, and even none of theese error conditions are checked (code assumes, that they never happen)

See also:

RFC 3629

5.1.3.3 void Scr::EncodeUTF8 (std::ostream & *o*, Uint *c*) throw ()

Parameters:

c character to encode

o reference to output stream

Print c directly to o in UTF8 encoded form

5.1.3.4 Uint Scr::StringLengthUTF8 (const char $*$ *s*) throw (Screen::InvalidUTF8)

Parameters:

s UTF-8 string

Compute length of null-terminated utf-8 string, that is number of UNICODE characters, not number of bytes in UTF-8 encoded version.

Note:

function assumes, that string is correct. No validation or range checking is performed

5.1.3.5 unsigned long Scr::width (wchar_t *c*) `[inline]`

Computes width of unicode character (0 or 1 or 2), that means number of cells in console, it needs to fit. Furthermore, it returns -1 if a character is a non-printable one.

5.2 Scr::Bg Namespace Reference

Background colors. WITHOUT style.

Enumerations

- enum Color {
  System = 0, Transparent = 1, Black = 40, Red = 41,
  Green = 42, Yellow = 43, Blue = 44, Magenta = 45,
  Cyan = 46, White = 47 }

  background colours enumeration

5.2.1 Detailed Description

Background colors. WITHOUT style.

5.2.2 Enumeration Type Documentation

5.2.2.1 enum Scr::Bg::Color

background colours enumeration

Enumerator:

System special colour represents default colour of system (for some terminals and terminal emulator this may differ from 8 basic colors)

Transparent Set colour of just-replaced text.

Black color 1

Red color 2

Green color 3

Yellow color 4

Blue color 5

Magenta color 6

Cyan color 7

White color 8

5.3 Scr::Control Namespace Reference

namespace containing iomanipulator-like items

Classes

- class _PositionYX

Enumerations

- enum _Refresh { Refresh }
- enum _Clear { Clear }

Functions

- _PositionYX GotoYX (Uint _y, Uint _x)

5.3.1 Detailed Description

namespace containing iomanipulator-like items

5.3.2 Enumeration Type Documentation

5.3.2.1 enum Scr::Control::_Clear

Special one-element type introduced only for Clear manipulator

Enumerator:

Clear This manipulator clears whole screen. FooScreen << Scr::Control::Clear is an direct equivalent of FooScreen.Clear().

5.3.2.2 enum Scr::Control::_Refresh

Special one-element type introduced only for Refresh manipulator

Enumerator:

Refresh This manipulator forces refreshing of screen. FooScreen << Scr::Control::Refresh is an direct equivalent of FooScreen.Refresh().

5.3.3 Function Documentation

5.3.3.1 Control::_PositionYX Scr::Control::GotoYX (Uint *_y*, Uint *_x*)

Parameters:

_y row on screen

_x column on screen

Controlling screen active point position (the point, where text starts). FooScreen << Scr::Control::GotoYX(3,4) is an direct equivalent of FooScreen.GotoYX(3,4).

5.4 Scr::Fg Namespace Reference

Foreground colors and styles.

Enumerations

- enum Color {
  System = 0, Transparent = 1, Black = 30, Red = 31,
  Green = 32, Yellow = 33, Blue = 34, Magenta = 35,
  Cyan = 36, White = 37 }

  Color itself. 8 basic colours + 2 special (Fg::System, Fg::Transparent).
- enum Style

  foreground styles

5.4.1 Detailed Description

Foreground colors and styles.

5.4.2 Enumeration Type Documentation

5.4.2.1 enum Scr::Fg::Color

Color itself. 8 basic colours + 2 special (Fg::System, Fg::Transparent).

Enumerator:

System special colour represents default colour of system (for some terminals and terminal emulator this may differ from 8 basic colors)

Transparent special colour represents colour of just-replaced character

Black color 1

Red color 2

Green color 3

Yellow color 4

Blue color 5

Magenta color 6

Cyan color 7

White color 8

5.5 Scr::TI Namespace Reference

Classes

- class TerminfoEntry

  Terminfo entry for single terminal type.

- class TerminfoCore

  Terminfo subsystem core: manages entries etc.

- class Keymap

  Class responsible for mapping control sequences to unique key codes.

- class TerminfoDatabase

  terminfo database finds system database and fetches entries

Enumerations

- enum Booleans
- enum Numbers
- enum Strings

5.5.1 Detailed Description

Terminfo database connectivity facilities

5.5.2 Enumeration Type Documentation

5.5.2.1 enum Scr::TI::Booleans

ordering of booleans in compiled terminfo file. This is based on /usr/include/term.h, by Zeyd M. Ben-Halim, Eric S. Raymond and Thomas E. Dickey.

5.5.2.2 enum Scr::TI::Numbers

ordering of numbers in compiled terminfo file. This is based on /usr/include/term.h, by Zeyd M. Ben-Halim, Eric S. Raymond and Thomas E. Dickey.

5.5.2.3 enum Scr::TI::Strings

ordering of strings in compiled terminfo file. This is based on /usr/include/term.h, by Zeyd M. Ben-Halim, Eric S. Raymond and Thomas E. Dickey.

5.6 Scr::Tk::Detail Namespace Reference

Selection form widget.

5.6.1 Detailed Description

Selection form widget.

5.7 TELNET Namespace Reference

Telnet control codes.

Variables

- const unsigned char BINARY = 0x00

  Binary mode.

- const unsigned char ECHO = 0x01

  Local/remote echo mode.

- const unsigned char SGA = 0x03

  Suppress go ahead.

- const unsigned char TTYPE = 0x18

  Terminal Type negotiation.

- const unsigned char SEND = 0x01

  request terminal type information

- const unsigned char IS = 0x00

  inform about terminal type

- const unsigned char NAWS = 0x1F

  Negotiate about Window Size.

- const unsigned char LINEMODE = 0x24

  Line mode negotiation.

- const unsigned char SE = 0xF0

  Subnegotiation end.

- const unsigned char NOP = 0xF1

  No operation.

- const unsigned char DM = 0xF2

  Data mark.

- const unsigned char BRK = 0xF3

  Break.

- const unsigned char IP = 0xF4

  Interrupt Process.

- const unsigned char AO = 0xF5

  Abort Output.

- const unsigned char AYT = 0xF6

  Are you there?

- const unsigned char EC = 0xF7

  Erase character.

- const unsigned char EL = 0xF8

  Erase line.

- const unsigned char GA = 0xF9

  Go ahead (allow other end to transmit).

- const unsigned char SB = 0xFA

  Subnegotiation begin.

- const unsigned char WILL = 0xFB

  Will (meaning depends on feature, we negotiate).

- const unsigned char WONT = 0xFC

  Won't (meaning depends on feature, we negotiate).

- const unsigned char DO = 0xFD

  Do (meaning depends on feature, we negotiate).

- const unsigned char DONT = 0xFE

  Don't (meaning depends on feature, we negotiate).

- const unsigned char IAC = 0xFF

  Interpret as command.

5.7.1 Detailed Description

Telnet control codes.

Whole set of constants useful for telnet negotiations as server or client. All of them are declared in apropriate RFC's.

5.7.2 Variable Documentation

5.7.2.1 const unsigned char TELNET::ECHO = 0x01

Local/remote echo mode.

IAC WILL ECHO sent by server disables local echo

See also:

RFC 857

5.7.2.2 const unsigned char TELNET::IAC = 0xFF

Interpret as command.

Special code in the beginning of all control sequences.

5.7.2.3 const unsigned char TELNET::IS = 0x00

inform about terminal type

Command code used by client while informing about terminal type during TTYPE subnegotiation

See also:

RFC 1091

5.7.2.4 const unsigned char TELNET::LINEMODE = 0x24

Line mode negotiation.

For description of this feature refer to appropriate RFC

See also:

RFC 1184

5.7.2.5 const unsigned char TELNET::NAWS = 0x1F

Negotiate about Window Size.

See also:

RFC 1073

5.7.2.6 const unsigned char TELNET::NOP = 0xF1

No operation.

Do not do anything

5.7.2.7 const unsigned char TELNET::SE = 0xF0

Subnegotiation end.

Special code inserted at the end of subnegotiation block

5.7.2.8 const unsigned char TELNET::SEND = 0x01

request terminal type information

Command code used by server while requesting TTYPE

See also:

RFC 1091

5.7.2.9 const unsigned char TELNET::SGA = 0x03

Suppress go ahead.

See also:

RFC 858

5.7.2.10 const unsigned char TELNET::TTYPE = 0x18

Terminal Type negotiation.

Detect terminal type and - possibly - detect it's additional emulation modes and switch between them. Documentation for this feature described in appropriate RFC.

See also:

RFC 1091

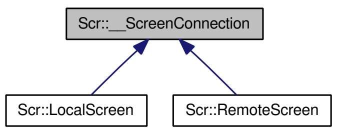
6 Class Documentation

6.1 Scr::__ScreenConnection Class Reference

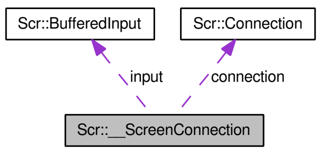
internal class which is base for all connection-specific implementations of screen (multiple-inheritance case)

`#include <screenconnection.h++>`

Inheritance diagram for Scr::__ScreenConnection:



Collaboration diagram for Scr::__ScreenConnection:



Public Member Functions

- __ScreenConnection (Connection &_connection, std::istream &_input) throw ()

- virtual int ProcessConnection ()=0
- virtual void ExitConnection (int _code)
- __ScreenConnection (Connection &_connection, std::istream &_input) throw ()
- virtual int ProcessConnection ()=0
- virtual void ExitConnection (int _code)

Protected Member Functions

- virtual Key DecodeKeyPressed ()

Protected Attributes

- int exitcode
- Connection & connection
- bool active

6.1.1 Detailed Description

internal class which is base for all connection-specific implementations of screen (multiple-inheritance case)

It represents internal interface between Scr::Connection and Scr::Screen classes.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Scr::__ScreenConnection::__ScreenConnection (Connection & *_connection*, std::istream & *_input*) throw ()

Parameters:

_input input stream (used to capture some of events, in particular keyboard events).

_connection newely estabilished connection to serve

6.1.2.2 Scr::__ScreenConnection::__ScreenConnection (Connection & *_connection*, std::istream & *_input*) throw ()

Parameters:

_input input stream (used to capture some of events, in particular keyboard events).

_connection newely estabilished connection to serve

6.1.3 Member Function Documentation

6.1.3.1 virtual Key Scr::__ScreenConnection::DecodeKeyPressed () `[protected, virtual]`

get key esc-code from std input stream. decode it into form from keyboard.h++

6.1.3.2 virtual int Scr::__ScreenConnection::ProcessConnection () `[pure virtual]`

Returns:

value of exitcode, as it was in the moment of connection termination if successful.

Initialize, conduct and end connection in way apropriate to connection type, operating system etc. Inform Scr::Connection object supplied about all captured events

Note:

as function (for design reasons) lacks exception-set specification, it may throw any exceptions, but it is recommended, that only exceptions typical to Scr::Connection::Start() will be thrown.

Implemented in Scr::LocalScreen, and Scr::RemoteScreen.

6.1.3.3 void Scr::__ScreenConnection::ExitConnection (int *_code*) `[virtual]`

Parameters:

_code exit code return from ProcessConnection after successfully finished connection

Force stopping connection as soon as possible

Note:

as function (for design reasons) lacks exception-set specification, it may throw any exceptions, but it is recommended, that only exceptions typical to Scr::Connection::Exit() will be thrown.

6.1.3.4 virtual int Scr::__ScreenConnection::ProcessConnection () `[pure virtual]`

Returns:

value of exitcode, as it was in the moment of connection termination if successful.

Initialize, conduct and end connection in way apropriate to connection type, operating system etc. Inform Scr::Connection object supplied about all captured events

Note:

as function (for design reasons) lacks exception-set specification, it may throw any exceptions, but it is recommended, that only exceptions typical to Scr::Connection::Start() will be thrown.

Implemented in Scr::LocalScreen, and Scr::RemoteScreen.

6.1.3.5 virtual void Scr::__ScreenConnection::ExitConnection (int *_code*) `[virtual]`

Parameters:

_code exit code return from ProcessConnection after successfully finished connection

Force stopping connection as soon as possible

Note:

as function (for design reasons) lacks exception-set specification, it may throw any exceptions, but it is recommended, that only exceptions typical to Scr::Connection::Exit() will be thrown.

6.1.4 Member Data Documentation

6.1.4.1 int Scr::__ScreenConnection::exitcode `[protected]`

ProcessConnection will return this value upon successful finish

6.1.4.2 Connection & Scr::__ScreenConnection::connection `[protected]`

is application running? does it have to stop? (ExitConnection() is called by Connection::Exit(int), sets exit code and breaks main loop performed in ProcessConnection)

6.1.4.3 bool Scr::__ScreenConnection::active `[protected]`

break main loop if set to false

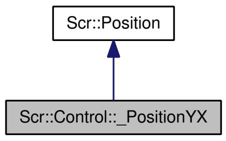
The documentation for this class was generated from the following files:

- lib/screen/include/connection.h++
- lib/screen/include/screenconnection.h++
- lib/screen/src/real/screenconnection.c++

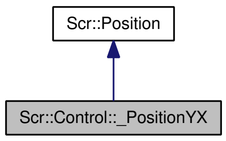
6.2 Scr::Control::_PositionYX Class Reference

`#include <screen.h++>`

Inheritance diagram for Scr::Control::_PositionYX:



Collaboration diagram for Scr::Control::_PositionYX:



6.2.1 Detailed Description

This is „private" class of system. It is only designed as a return type of Scr::Control::GotoYX(Uint , Uint) - simmilar idea to std::_Setw (as return type of std::setw(int)).

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

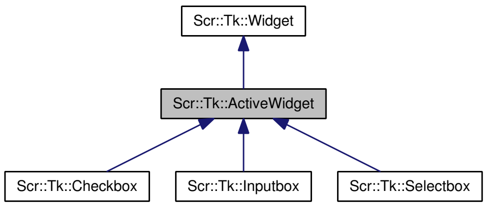
- include/rexio/screen.h++

6.3 Scr::Tk::ActiveWidget Class Reference

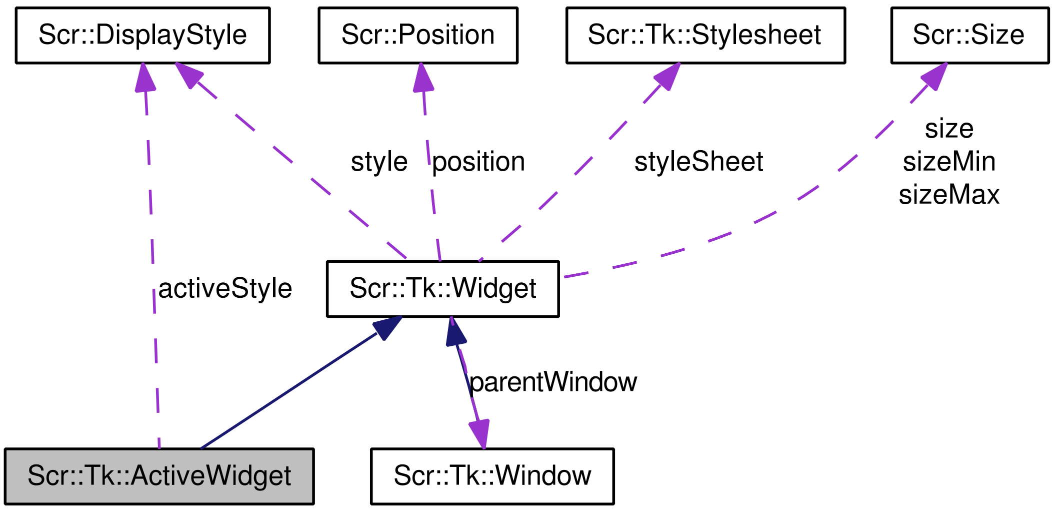
Focus capable widget.

`#include <activewidget.h++>`

Inheritance diagram for Scr::Tk::ActiveWidget:



Collaboration diagram for Scr::Tk::ActiveWidget:



Public Member Functions

- void OnFocus (FocusPolicy focustype) throw ()
- void OnUnFocus (FocusPolicy focustype) throw ()
- void OnKeyDown (Key key) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Member Functions

- virtual void SetStylesheet (Stylesheet ∗_styleSheet) throw ()

6.3.1 Detailed Description

Focus capable widget.

Focusable widget, useful for input fields and other form elements.

6.3.2 Member Function Documentation

6.3.2.1 virtual void Scr::Tk::ActiveWidget::SetStylesheet (Stylesheet ∗ *_styleSheet*) throw () `[inline, protected, virtual]`

Parameters:

_styleSheet pointer to style data

Apply Stylesheet to this widget. Reinitialize any style properties if their alternatives are supplied.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Inputbox.

6.3.2.2 void ActiveWidget::OnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element focused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Selectbox.

6.3.2.3 void ActiveWidget::OnUnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element unfocused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Selectbox.

6.3.2.4 void ActiveWidget::OnKeyDown (Key *key*) throw () `[virtual]`

Parameters:

key keycode

Keyboard button press event.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Inputbox.

6.3.2.5 virtual bool Scr::Tk::ActiveWidget::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Checkbox, Scr::Tk::Inputbox, and Scr::Tk::Selectbox.

6.3.2.6 virtual const char∗ Scr::Tk::ActiveWidget::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Checkbox, Scr::Tk::Inputbox, and Scr::Tk::Selectbox.

6.3.2.7 virtual const char∗ Scr::Tk::ActiveWidget::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Checkbox, Scr::Tk::Inputbox, and Scr::Tk::Selectbox.

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

- include/rexio/tk/activewidget.h++
- lib/toolkit/src/activewidget.c++

6.4 Scr::AutoList< T > Class Template Reference

container combining advantages of list and hash map, allowing

`#include <autolist.h++>`

Public Member Functions

- iterator operator[ ] (T &elem)
- reverse_iterator rbegin ()
- reverse_iterator rend ()
- iterator begin ()
- iterator end ()

- const_iterator begin () const
- const_iterator end () const
- const T & back ()
- size_type size ()
- bool empty ()
- void erase (iterator i)
- void remove (T elem)
- iterator insert (const T &before, const T &newelem)
- void push_front (const T &elem)
- void push_back (const T &elem)
- void swap (const T &elem1, const T &elem2)

6.4.1 Detailed Description

template<class T> class Scr::AutoList< T >

container combining advantages of list and hash map, allowing

It is implemented using standard STL list and almost_standard hash_map.

6.4.2 Member Function Documentation

6.4.2.1] template<class T> iterator **Scr::AutoList**< T >::operator[ ] (T & *elem*) `[inline]`

Parameters:

elem element to find

Returns:

list iterator to specific element

6.4.2.2 template<class T> reverse_iterator Scr::AutoList< T >::rbegin () `[inline]`

Returns:

list iterator to lase element

6.4.2.3 template<class T> reverse_iterator Scr::AutoList< T >::rend () `[inline]`

Returns:

list iterator rend() of list

6.4.2.4 template<class T> iterator Scr::AutoList< T >::begin () `[inline]`

Returns:

list iterator to first element

6.4.2.5 template<class T> iterator Scr::AutoList< T >::end () `[inline]`

Returns:

list iterator end() of list

6.4.2.6 template<class T> const_iterator Scr::AutoList< T >::begin () const `[inline]`

Returns:

list iterator to first element

6.4.2.7 template<class T> const_iterator Scr::AutoList< T >::end () const `[inline]`

Returns:

list iterator end() of list

6.4.2.8 template<class T> const T& Scr::AutoList< T >::back () `[inline]`

Returns:

last element in the list

6.4.2.9 template<class T> size_type Scr::AutoList< T >::size () `[inline]`

Returns:

number of elements

6.4.2.10 template<class T> bool Scr::AutoList< T >::empty () `[inline]`

Returns:

true if _size is 0

6.4.2.11 template<class T> void Scr::AutoList< T >::erase (iterator *i*) `[inline]`

Parameters:

i list iterator to specific element to be erased

6.4.2.12 template<class T> void Scr::AutoList< T >::remove (T *elem*) `[inline]`

Parameters:

elem specific element to be erased

6.4.2.13 template<class T> iterator Scr::AutoList< T >::insert (const T & *before*, const T & *newelem*) `[inline]`

Parameters:

before where to insert

newelem what to insert

6.4.2.14 template<class T> void Scr::AutoList< T >::push_front (const T & *elem*) `[inline]`

Parameters:

elem what to insert

6.4.2.15 template<class T> void Scr::AutoList< T >::push_back (const T & *elem*) `[inline]`

Parameters:

elem what to insert

6.4.2.16 template<class T> void Scr::AutoList< T >::swap (const T & *elem1*, const T & *elem2*) `[inline]`

Parameters:

elem1 element to be swapped w/ elem2

elem2 element to be swapped w/ elem1

swaps two elements in the Autolist

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

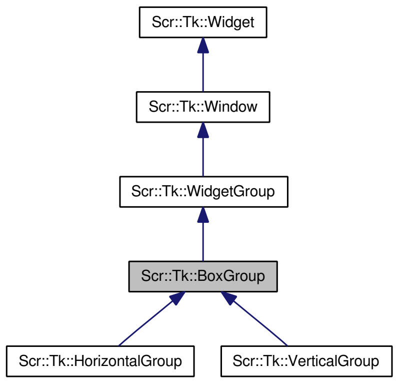
- include/rexio/tk/autolist.h++

6.5 Scr::Tk::BoxGroup Class Reference

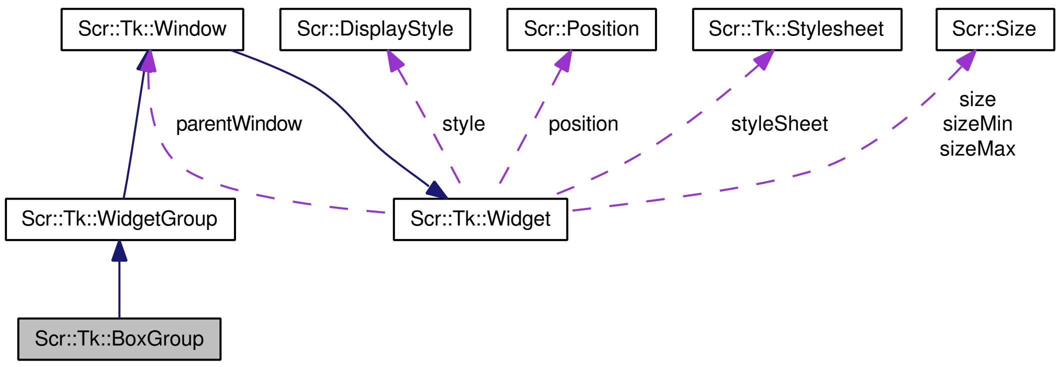
Provides horizontal and vertical widget grouping capabilities.

```
#include <boxgroup.h++>
```

Inheritance diagram for Scr::Tk::BoxGroup:



Collaboration diagram for Scr::Tk::BoxGroup:



Public Types

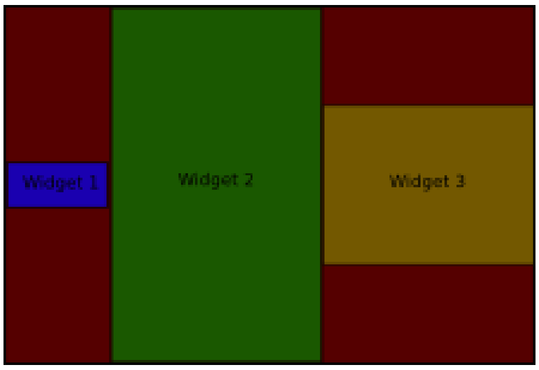
- enum AlignPolicy { Begin, Center, End, Distribute }

Public Member Functions

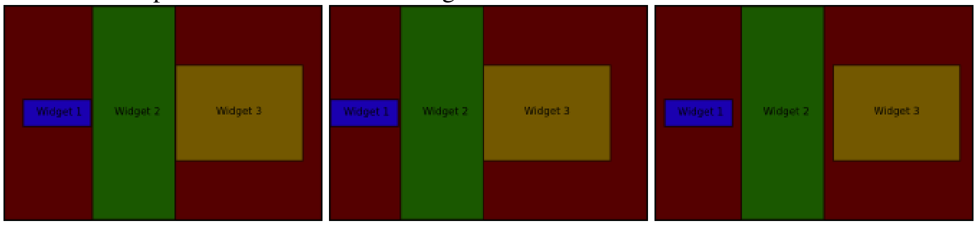
- virtual void SwapWidgets (Widget &widget1, Widget &widget2) throw ()
- virtual void AddWidget (Widget &widget) throw ()
- virtual void AddWidget (Widget &widget, Uint stretchFactor) throw ()
- virtual void DelWidget (Widget &widget) throw ()
- virtual void OnResize () throw ()
- virtual void SetAlignPolicy (AlignPolicy _alignPolicy) throw ()
- virtual AlignPolicy GetAlignPolicy () throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char $*$ TypeName () const
- virtual const char $*$ ParentName () const

Protected Member Functions

- virtual void ArrangeContents ()=0 throw ()



BoxGroup in Horizontal Mode. Widget1's stretchFactor == 1 and the others' is 2.



All of the widgets here have their maxSize set so that there is still free space.
It shows different types of AlignPolicy. Respectively: Center, Start, Distributed.

Protected Attributes

- std::tr1::unordered_map< const Widget ∗, LayoutData, _hash > elementsLayout
- AlignPolicy alignPolicy

Classes

- struct LayoutData

  Additional data used for positioning.

6.5.1 Detailed Description

Provides horizontal and vertical widget grouping capabilities.

Intelligently places the containing widgets among allocated space. Widgets can be placed vertically or horizontally.

See also:

VerticalGroup and *HorizontalGroup* provided for convenience.

6.5.2 Member Enumeration Documentation

6.5.2.1 enum Scr::Tk::BoxGroup::AlignPolicy

Widget aligning policy in case of not all space being used.

Enumerator:

Begin Align everything to the left/top depending on *groupType*.

Center Align everything to the center.

End Align everything to the right/bottom depending on *groupType*.

Distribute Try to evenly distribute free space between widgets, adding a margin between each of them.

6.5.3 Member Function Documentation

6.5.3.1 virtual void Scr::Tk::BoxGroup::ArrangeContents () throw () `[protected, pure virtual]`

where all magic is done :)

Reimplemented from Scr::Tk::WidgetGroup.

Implemented in Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

6.5.3.2 void BoxGroup::SwapWidgets (Widget & *widget1*, Widget & *widget2*) throw () `[virtual]`

Parameters:

widget1 First widget

widget2 Second widget

Swap two widgets with together, provided that they are being contained by the WidgetGroup. rearrange contents afterwards

Reimplemented from Scr::Tk::WidgetGroup.

6.5.3.3 void BoxGroup::AddWidget (Widget & *widget*) throw () `[virtual]`

Parameters:

widget widget to attach to this window

Attach a widget to this window. Specifically, add it to the *elements*.

Exceptions:

ParentAlreadySet is thrown if the widget has already been attached to some other window.

WidgetAlreadyAdded if the widget is already attached to THIS window.

Reimplemented from Scr::Tk::Window.

6.5.3.4 void BoxGroup::AddWidget (Widget & *widget*, Uint *stretchFactor*) throw () `[virtual]`

Parameters:

widget widget to attach to this window

stretchFactor part of the added widget's *LayoutData*

Attach a widget to this window. Specifically, add it to the *elements*.

6.5.3.5 void BoxGroup::DelWidget (Widget & *widget*) throw () `[virtual]`

Parameters:

widget widget to detach from this window

Detach a widget from this window. Specifically, del it from the *elements*.

Exceptions:

WidgetNotPresent is thrown if the widget is not attached to this window.

Reimplemented from Scr::Tk::Window.

6.5.3.6 void BoxGroup::OnResize () throw () `[virtual]`

Resize event. Do something i.e. adjust content to the new size.

Reimplemented from Scr::Tk::Window.

6.5.3.7 void BoxGroup::SetAlignPolicy (AlignPolicy *_alignPolicy*) throw () `[virtual]`

Parameters:

_alignPolicy enumerative type parameter specifying aling policy (refer to documentation for this class for information on it)

Set new BoxGroupType. Can be invoked anytime and it will initiate a redraw.

6.5.3.8 BoxGroup::AlignPolicy BoxGroup::GetAlignPolicy () throw () `[virtual]`

Get current AlignPolicy.

6.5.3.9 virtual bool Scr::Tk::BoxGroup::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::WidgetGroup.

Reimplemented in Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

6.5.3.10 virtual const char* Scr::Tk::BoxGroup::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::WidgetGroup.

Reimplemented in Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

6.5.3.11 virtual const char* Scr::Tk::BoxGroup::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::WidgetGroup.

Reimplemented in Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

6.5.4 Member Data Documentation

6.5.4.1 std::tr1::unordered_map<const Widget*, LayoutData,_hash> Scr::Tk::BoxGroup::elementsLayout `[protected]`

Associates *LayoutData* to each attached widget.

6.5.4.2 AlignPolicy Scr::Tk::BoxGroup::alignPolicy `[protected]`

Current aligning policy.

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

- include/rexio/tk/boxgroup.h++
- lib/toolkit/src/boxgroup.c++

6.6 Scr::Tk::BoxGroup::LayoutData Struct Reference

Additional data used for positioning.

`#include <boxgroup.h++>`

Public Attributes

- Uint stretchFactor

6.6.1 Detailed Description

Additional data used for positioning.

Widget layouting information inside BoxGroup.

6.6.2 Member Data Documentation

6.6.2.1 Uint Scr::Tk::BoxGroup::LayoutData::stretchFactor

Defines a factor of dividing free space between widgets. i.e. space = (this_factor/sum_of_factors) * freespace.

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

- include/rexio/tk/boxgroup.h++

6.7 Scr::BufferedInput Class Reference

Intermediate between Scr::__ScreenConnection and std::istream.

```
#include <connection.h++>
```

Public Member Functions

- BufferedInput (std::istream &_stream) throw ()
- void Buffer () throw ()
- bool HasBufferedText () const throw ()
- unsigned char TryPeek () const throw (BufferEmpty)
- unsigned char TryGet () throw (BufferEmpty)
- unsigned char Peek () const throw ()
- unsigned char Get () throw ()
- void UnGet () throw (BufferEmpty)
- int FD () const throw ()
- const std::istream & Stream () const throw ()
- std::istream & Stream () throw ()
- std::string String () throw ()
- std::string DebugInfo () throw ()
- const std::string DebugInfo () const throw ()
- bool KbHit () throw ()
- BufferedInput (std::istream &_stream) throw ()
- void Buffer () throw ()
- bool HasBufferedText () throw ()
- unsigned char Peek () throw (BufferEmpty)
- unsigned char Get () throw (BufferEmpty)
- void UnGet () throw ()
- int FD () throw ()
- std::istream & Stream () throw ()

Private Member Functions

- void ForceBuffer () const throw ()
- void ForceBuffer () throw ()

Private Attributes

- bool filledToCapacity
- Uint currentCharBufferSize
- Uint currentCharBufferIndex
- char charBuffer [maxCharBufferSize]
- std::istream & stream

6.7.1 Detailed Description

Intermediate between Scr::__ScreenConnection and std::istream.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Scr::BufferedInput::BufferedInput (std::istream & *_stream*) throw () [inline, explicit]

Parameters:

_stream stream to be contained

6.7.2.2 Scr::BufferedInput::BufferedInput (std::istream & *_stream*) throw () [inline, explicit]

Parameters:

_stream stream to be contained

6.7.3 Member Function Documentation

6.7.3.1 void BufferedInput::ForceBuffer () const throw () [private]

std::istream::readsome returned 0, while something needs to be read.

6.7.3.2 void Scr::BufferedInput::Buffer () throw () [inline]

Save some characters in internal buffer (it is not invoked automatically when Get() is called and buffer is empty.

6.7.3.3 bool Scr::BufferedInput::HasBufferedText () const throw () [inline]

Inquiry if object has some buffered text, or at least can make this text availble instantly

6.7.3.4 unsigned char Scr::BufferedInput::TryPeek () const throw (BufferEmpty) [inline]

Peek if it won't block app

6.7.3.5 unsigned char Scr::BufferedInput::TryGet () throw (BufferEmpty) [inline]

Get if it won't block app (throw exception otherwise)

6.7.3.6 unsigned char Scr::BufferedInput::Peek () const throw () [inline]

Current character. The same will be availble after call to this func.

6.7.3.7 unsigned char Scr::BufferedInput::Get () throw () [inline]

Get character from stream

6.7.3.8 void Scr::BufferedInput::UnGet () throw (BufferEmpty) [inline]

UnGet().

Note:

that this function may fail if called just after buffering, or called too frequently: only one successful UnGet per one Get is guaranteed.

Exceptions:

Scr::BufferedString::BufferEmpty is thrown when too many UnGet's are processed oneafter another

6.7.3.9 int Scr::BufferedInput::FD () const throw () `[inline]`

Unix style file descriptor

6.7.3.10 const std::istream& Scr::BufferedInput::Stream () const throw () `[inline]`

direct access to underlying std::stream - const version

6.7.3.11 std::istream& Scr::BufferedInput::Stream () throw () `[inline]`

direct access to underlying std::stream

6.7.3.12 string BufferedInput::String () throw ()

contents of internal buffer as string

6.7.3.13 string BufferedInput::DebugInfo () throw ()

more than info returned by String(): function created specifically for debugging/logging purposes

6.7.3.14 const string BufferedInput::DebugInfo () const throw ()

more than info returned by String(): function created specifically for debugging/logging purposes

6.7.3.15 void Scr::BufferedInput::ForceBuffer () throw () `[private]`

Blocking buffering function

6.7.3.16 bool Scr::BufferedInput::KbHit () throw ()

Returns:

true if input device is ready to transmit data

6.7.3.17 void Scr::BufferedInput::Buffer () throw () `[inline]`

Save some characters in internal buffer (it is not invoked automatically when Get() is called and buffer is empty.

6.7.3.18 bool Scr::BufferedInput::HasBufferedText () throw () `[inline]`

Returns:

true if any character is availble in buffer

6.7.3.19 unsigned char Scr::BufferedInput::Peek () throw (BufferEmpty) [inline]

Returns:

first availble character w/o moving pointer

6.7.3.20 unsigned char Scr::BufferedInput::Get () throw (BufferEmpty) [inline]

get character

6.7.3.21 void Scr::BufferedInput::UnGet () throw () [inline]

Returns:

character to buffer

6.7.3.22 int Scr::BufferedInput::FD () throw () [inline]

Returns:

UNIX∗ file descriptor for associated stream.

∗ UNIX is registered trademark of AT&T and OpenGroup.

6.7.3.23 std::istream& Scr::BufferedInput::Stream () throw () [inline]

Returns:

associated C++ stream.

6.7.4 Member Data Documentation

6.7.4.1 bool Scr::BufferedInput::filledToCapacity [mutable, private]

if after last read buffer was filled while still something on input was availble

6.7.4.2 Uint Scr::BufferedInput::currentCharBufferSize [mutable, private]

number of characters staying in buffer after last read.

6.7.4.3 Uint Scr::BufferedInput::currentCharBufferIndex [mutable, private]

idx of current character

6.7.4.4 char Scr::BufferedInput::charBuffer [mutable, private]

read some bytes from input, then transform em to keyboard events (no direct access to istream outside of ProcessConnection, where readsome() performed - to ensure)

6.7.4.5 std::istream & Scr::BufferedInput::stream `[mutable, private]`

input stream

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

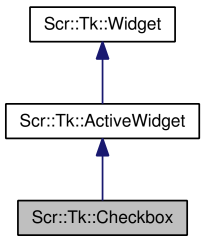
- lib/screen/include/bufferedinput.h++
- lib/screen/include/connection.h++
- lib/screen/src/core/bufferedinput.c++

6.8 Scr::Tk::Checkbox Class Reference

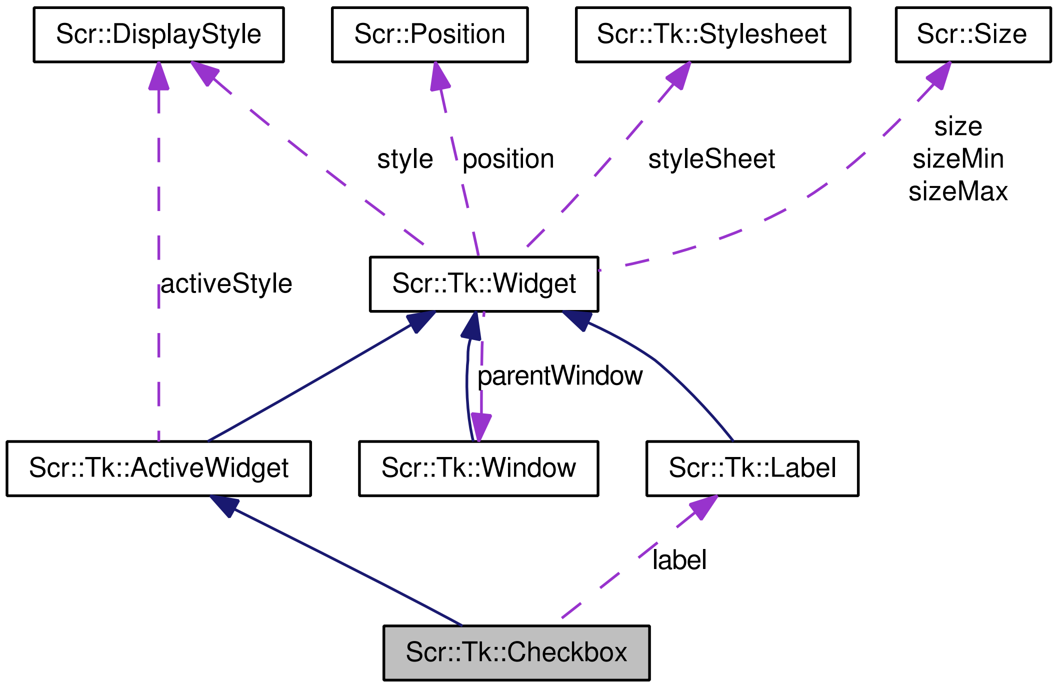
two-state widget

`#include <checkbox.h++>`

Inheritance diagram for Scr::Tk::Checkbox:



Collaboration diagram for Scr::Tk::Checkbox:



Public Member Functions

- void OnRedraw (Screen &screen) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Private Attributes

- Label label

  label near the checkbox

- bool state

6.8.1 Detailed Description

two-state widget

A widgets that indicates setting of boolean feature canonly be turned on and off. It has label, that indicates its name and boolean field that indicates its current state

6.8.2 Member Function Documentation

6.8.2.1 void Checkbox::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Widget.

6.8.2.2 virtual bool Scr::Tk::Checkbox::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::ActiveWidget.

6.8.2.3 virtual const char∗ Scr::Tk::Checkbox::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::ActiveWidget.

6.8.2.4 virtual const char∗ Scr::Tk::Checkbox::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::ActiveWidget.

6.8.3 Member Data Documentation

6.8.3.1 bool Scr::Tk::Checkbox::state `[private]`

current state (on/off) displayed to user

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

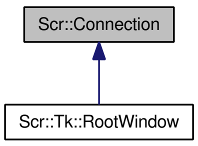
- include/rexio/tk/checkbox.h++
- lib/toolkit/src/checkbox.c++

6.9 Scr::Connection Class Reference

Class representing basic input and output operations.

`#include <screen.h++>`

Inheritance diagram for Scr::Connection:



Public Member Functions

- virtual int Start () throw (StartFailed,Screen::IllegalCharacter)
- virtual int Start (int argc, char ∗∗argv) throw (StartFailed,Screen::IllegalCharacter)
- void Exit (int code) throw (StopFailed)
- virtual void OnExit (int code) throw ()

6.9.1 Detailed Description

Class representing basic input and output operations.

Class is implemented and designed as base class for any specific application. It controls directly screen size, and specifies event interface for reacting keyboard and screen connected events. It is designed to be platform-transparent, so programmer does not have to bother OS specific method of checking window size, key value etc.

OnEvent actions are defined as virtual member functions

6.9.2 Member Function Documentation

6.9.2.1 int Scr::Connection::Start () throw (StartFailed,Screen::IllegalCharacter) `[virtual]`

Returns:

result of whole connection. If broken, the result is 1. Else the result is argument passed to Exit(int)

Start connection (with no arguments - they must be set with application specific methods defined by programmer). Function blocks execution of thread up to finish of connection.

Exceptions:

Scr::Connection::AlreadyRunning when connection has already been started (one execution thread per class instance allowed) and hasn't yet been stopped.

Scr::Connection::Broken is thrown when connection is broken (i.e. input/output error occured)

Scr::Connection::FailedToStart when connecction can not be estabilished for some reason.

Note:

as Start() is defined in way, that allows it to throw only one exception class and all OnEvent functions do not allow any exceptions, all of them must be handled within exception handling function. Unexpected exception handler will be used otherwise.

Reimplemented in Scr::Tk::RootWindow.

6.9.2.2 int Scr::Connection::Start (int *argc*, char $**$ *argv*) throw (StartFailed,Screen::IllegalCharacter) `[virtual]`

Parameters:

argc number of arguments

argv C-style array of arguments

Start connection. *argv* can be parsed in inheritting classes.

See also:

Start() for detailed info

Reimplemented in Scr::Tk::RootWindow.

6.9.2.3 void Scr::Connection::Exit (int *code*) throw (StopFailed)

Returns:

nothing

Parameters:

code this will be the result of ongoing Start()

If connection is currently running (that means, Start() member function of specific object is running) Exit tells it to break as soon as possible, call OnExit() and return code given.

Exceptions:

Scr::Connection::AlreadyStopped exception is thrown when Exit was already called, but connection wasn't stopped yet.

Scr::Connection::NotYetStarted is thrown when connection was already stopped or hasn't yet been started.

6.9.2.4 void Scr::Connection::OnExit (int *code*) throw () `[virtual]`

Parameters:

code exit code. Will be returned by Start just after finish of app.

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

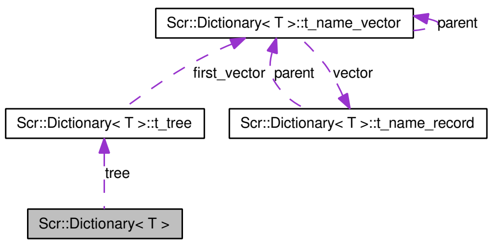
- include/rexio/screen.h++
- lib/screen/src/core/connection.c++

6.10 Scr::Dictionary< T > Class Template Reference

replacement of std::map<std::string,T> - optimized for string key random access using dictionary-tree data structure.

`#include <dictionary.h++>`

Collaboration diagram for Scr::Dictionary< T >:


Scr::Dictionary< T >::t_name_vector
parent
first_vector
parent
vector
Scr::Dictionary< T >::t_tree
Scr::Dictionary< T >::t_name_record
tree
Scr::Dictionary< T >

Protected Member Functions

- t_name_record ∗ tree_add (const char ∗name)
- t_name_record ∗ tree_partial_find (const char ∗name, t_name_vector ∗current_vector, size_t current=0) const
- t_name_record ∗ tree_find (const char ∗name, t_name_vector ∗current_vector, int current=0) const

Static Protected Member Functions

- static t_name_record ∗ tree_find_next (t_name_record ∗r)
- static void tree_erase_record (t_name_record ∗r)
- static void tree_erase_vector (t_name_vector ∗v)

Classes

- class iterator

  iterator class for Dictionary

- struct t_name_record

  tree leaf (node containing just one pc. of information

- struct t_name_vector

  node containing references to other nodes

- struct t_tree

  core information block (one per Dictionary)

6.10.1 Detailed Description

template<typename T> class Scr::Dictionary< T >

replacement of std::map<std::string,T> - optimized for string key random access using dictionary-tree data structure.

Member functions are named in C++ library convention, that is w/ underscore and w/o capital letters.

Note:

this class is not STL compatible. it is only STL-like.

6.10.2 Member Function Documentation

6.10.2.1 template<typename T> Dictionary< T >::t_name_record ∗ Scr::Dictionary< T >::tree_-add (const char ∗ *name*) `[inline, protected]`

add node to tree.

Parameters:

name name associated w/ node

Returns:

pointer to new node (or NULL if adding it was unsuccessful)

Exceptions:

std::bad_alloc if memory allocation failed

6.10.2.2 template<typename T> Dictionary< T >::t_name_record ∗ Scr::Dictionary< T >::tree_-partial_find (const char ∗ *name*, t_name_vector ∗ *current_vector*, size_t *current* = 0) const `[inline, protected]`

Attempts to search for a specific node. Doesn't modify tree (doesn't new node if search failed).

Returns:

Tf argument matches beginning of more than node key, t_name_vector is really returned (what may be detected by testing type member field), even if one of theese nodes matches completely. If nothing matches, 0 is returned. Otherwise ptr to record is returned

Parameters:

name key to look for

current_vector where to start search

current assume current depth in tree (start matching from this character of name)

6.10.2.3 template<typename T> Dictionary< T >::t_name_record ∗ Scr::Dictionary< T >::tree_-find (const char ∗ *name*, t_name_vector ∗ *current_vector*, int *current* = 0) const `[inline, protected]`

Returns:

pointer to specific node if it exists, NULL otherwise. this function depends on tree_partial_search

Parameters:

name key to look for

current_vector where to look for

current assume current depth in tree (start matching from this character of name)

6.10.2.4 template<typename T> Dictionary< T >::t_name_record ∗ Scr::Dictionary< T >::tree_-find_next (t_name_record ∗ *r*) `[inline, static, protected]`

Find next node. If r points to vector, find it's first node. If nothing found, return 0.

Parameters:

r record

6.10.2.5 template<typename T> static void Scr::Dictionary< T >::tree_erase_record (t_name_-record ∗ *r*) `[inline, static, protected]`

erase record

Parameters:

r record to be erased

6.10.2.6 template<typename T> void Scr::Dictionary< T >::tree_erase_vector (t_name_vector ∗ *v*) [inline, static, protected]

erase vector

Parameters:

v pointer to vector, that will be erased

Note:

function not only recursively erases contents of vector, but also erases vector itself

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

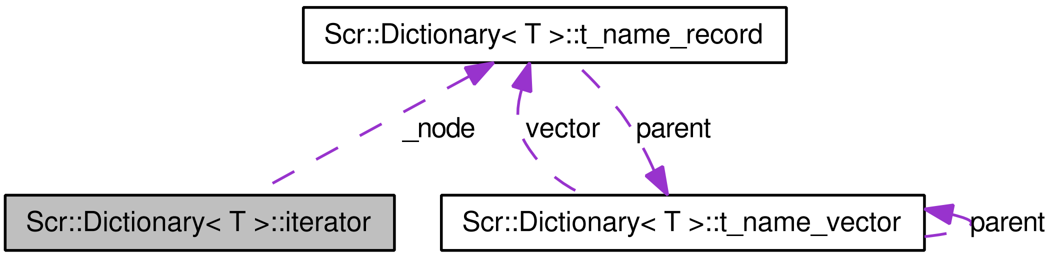
- lib/screen/include/dictionary.h++

6.11 Scr::Dictionary< T >::iterator Class Reference

iterator class for Dictionary

```
#include <dictionary.h++>
```

Collaboration diagram for Scr::Dictionary< T >::iterator:


Scr::Dictionary< T >::t_name_record
_node
vector
parent
Scr::Dictionary< T >::iterator
Scr::Dictionary< T >::t_name_vector
parent

Public Types

- enum validity { VALID, INVALID, NOT_UNIQUE, END }

  result of validity_test

Public Member Functions

- iterator ()
- iterator (const iterator &it)
- validity validity_test ()
- bool valid ()
- T & operator∗ ()
- T ∗ operator → ()
- iterator & operator= (const iterator &it)
- bool operator== (const iterator &it)
- bool operator!= (const iterator &it)
- bool operator< (const iterator &it)
- iterator & operator++ ()

Protected Member Functions

- iterator (t_name_record ∗__node)

6.11.1 Detailed Description

template<typename T> class Scr::Dictionary< T >::iterator

iterator class for Dictionary

6.11.2 Member Enumeration Documentation

6.11.2.1 template<typename T> enum Scr::Dictionary::iterator::validity

result of validity_test

Enumerator:

VALID	dereference (indirection) possible, iterator points to single data object
INVALID	unique key, but no data object (dereference WILL fail)
NOT_UNIQUE	not unique key: dreference WILL fail
END	end(): dreference WILL fail

6.11.3 Constructor & Destructor Documentation

6.11.3.1 template<typename T> Scr::Dictionary< T >::iterator::iterator () `[inline]`

Default constructor returns iterator, that equals end()

6.11.3.2 template<typename T> Scr::Dictionary< T >::iterator::iterator (const iterator & *it*) `[inline]`

copy constructor

Parameters:

it base of construction

6.11.3.3 template<typename T> Scr::Dictionary< T >::iterator::iterator (t_name_record ∗ *__node*) `[inline, explicit, protected]`

Constructor initialized w/ raw data node pointer (t_name_record) is accessed by functions such as begin(), end() or find().

Parameters:

__node node in tree mapped to this iterator

6.11.4 Member Function Documentation

6.11.4.1 template<typename T> validity Scr::Dictionary< T >::iterator::validity_test () `[inline]`

Tests if iterator is valid. If it is VALID is returned. if it is not, function says why

6.11.4.2 template<typename T> bool Scr::Dictionary< T >::iterator::valid () [inline]

tests if iterator is valid

6.11.4.3 template<typename T> T& Scr::Dictionary< T >::iterator::operator* () [inline]

Indirection operator returns reference to object

Exceptions:

std::bad_exception happens when iterator is not unique

6.11.4.4 template<typename T> T* Scr::Dictionary< T >::iterator::operator → () [inline]

Indirection-and-element-access operator returns reference to object

Exceptions:

std::bad_exception happens when iterator is not unique

6.11.4.5 template<typename T> iterator& Scr::Dictionary< T >::iterator::operator= (const iterator & *it*) [inline]

Assignment operator

Parameters:

it other iterator

6.11.4.6 template<typename T> bool Scr::Dictionary< T >::iterator::operator== (const iterator & *it*) [inline]

Comparison operator

Parameters:

it other iterator

6.11.4.7 template<typename T> bool Scr::Dictionary< T >::iterator::operator!= (const iterator & *it*) [inline]

Comparison operator

Parameters:

it other iterator

6.11.4.8 template<typename T> bool Scr::Dictionary< T >::iterator::operator< (const iterator & *it*) [inline]

tricky comparison operator comparing lexicographically w/ other key

Parameters:

it other iterator

6.11.4.9 template<typename T> iterator& Scr::Dictionary< T >::iterator::operator++ () `[inline]`

incrementation operator finds new element

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

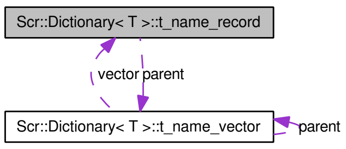
- lib/screen/include/dictionary.h++

6.12 Scr::Dictionary< T >::t_name_record Struct Reference

tree leaf (node containing just one pc. of information

`#include <dictionary.h++>`

Collaboration diagram for Scr::Dictionary< T >::t_name_record:


Scr::Dictionary< T >::t_name_record
vector
parent
Scr::Dictionary< T >::t_name_vector
parent

Public Attributes

- int type

    magic value to test, whenever it is a vector or a record

- char ∗ name

    key itself

- int num_occurrences

    number of occurences of specific key

6.12.1 Detailed Description

template<typename T> struct Scr::Dictionary< T >::t_name_record

tree leaf (node containing just one pc. of information

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

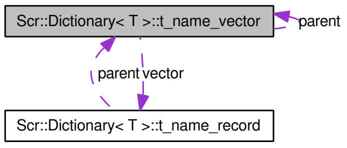
- lib/screen/include/dictionary.h++

6.13 Scr::Dictionary< T >::t_name_vector Struct Reference

node containing references to other nodes

`#include <dictionary.h++>`

Collaboration diagram for Scr::Dictionary< T >::t_name_vector:


Scr::Dictionary< T >::t_name_vector
parent
parent
vector
Scr::Dictionary< T >::t_name_record

Public Attributes

- int type

  magic value to test, whenever it is a vector or a record

6.13.1 Detailed Description

template<typename T> struct Scr::Dictionary< T >::t_name_vector

node containing references to other nodes

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

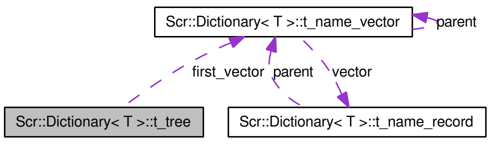
- lib/screen/include/dictionary.h++

6.14 Scr::Dictionary< T >::t_tree Struct Reference

core information block (one per Dictionary)

```
#include <dictionary.h++>
```

Collaboration diagram for Scr::Dictionary< T >::t_tree:


Scr::Dictionary< T >::t_name_vector
parent
first_vector
parent
vector
Scr::Dictionary< T >::t_tree
Scr::Dictionary< T >::t_name_record

Public Attributes

- int max_num_occurrences

  greatest recorded number of occurrences

- t_name_vector ∗ first_vector

  first vector

6.14.1 Detailed Description

template<typename T> struct Scr::Dictionary< T >::t_tree

core information block (one per Dictionary)

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

- lib/screen/include/dictionary.h++

6.15 Scr::DisplayStyle Class Reference

complete set of display properties for single character

```
#include <screen.h++>
```

Public Member Functions

- DisplayStyle (Fg::Color _fgColor, Fg::Style _fgStyle, Bg::Color _bgColor) throw ()
- DisplayStyle (const DisplayStyle &s) throw ()
- DisplayStyle () throw ()
- Fg::Color GetFgColor () const throw ()
- Fg::Style GetFgStyle () const throw ()
- Bg::Color GetBgColor () const throw ()
- void SetFgColor (const Fg::Color col) throw ()
- void SetFgStyle (const Fg::Style s) throw ()
- void SetBgColor (const Bg::Color col) throw ()
- bool operator== (const DisplayStyle &other)
- bool operator!= (const DisplayStyle &other)
- Scr::DisplayStyle & operator= (const DisplayStyle &other)

Private Attributes

- union {
  Uint32 style
  struct {
  unsigned char fgColor
  foreground color
  unsigned char fgStyle
  foreground style
  unsigned char bgColor
  background color
  } properties
  };

6.15.1 Detailed Description

complete set of display properties for single character

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Scr::DisplayStyle::DisplayStyle (Fg::Color *_fgColor*, Fg::Style *_fgStyle*, Bg::Color *_-bgColor*) throw ()

Set up specified style (parametrized constructor)

Parameters:

_fgColor

_fgStyle

_bgColor

6.15.2.2 Scr::DisplayStyle::DisplayStyle (const DisplayStyle & *s*) throw ()

Parameters:

s - source to copy

basic copy constructor - binary 1:1 copy.

6.15.2.3 Scr::DisplayStyle::DisplayStyle () throw ()

Nonparameter constructor sets colours default. default values are implementation-specific (currently white on green, but this may vary - maybe once upon the time we will implement some special "undefined" values for all three members of this class);

6.15.3 Member Function Documentation

6.15.3.1 Fg::Color Scr::DisplayStyle::GetFgColor () const throw () `[inline]`

Returns:

foreground color

6.15.3.2 Fg::Style Scr::DisplayStyle::GetFgStyle () const throw () `[inline]`

Returns:

foreground style

6.15.3.3 Bg::Color Scr::DisplayStyle::GetBgColor () const throw () `[inline]`

Returns:

nackground color

6.15.3.4 void Scr::DisplayStyle::SetFgColor (const Fg::Color *col*) throw () `[inline]`

Set foreground color

Parameters:

col new color

6.15.3.5 void Scr::DisplayStyle::SetFgStyle (const Fg::Style *s*) throw () `[inline]`

Set foreground style

Parameters:

s new style

6.15.3.6 void Scr::DisplayStyle::SetBgColor (const Bg::Color *col*) throw () `[inline]`

Set background color

Parameters:

col new color

6.15.3.7 bool Scr::DisplayStyle::operator== (const DisplayStyle & *other*) `[inline]`

Parameters:

other (fgColor==other.fgColor) && (fgStyle==other.fgStyle) && (bgColor==other.bgColor)

6.15.3.8 bool Scr::DisplayStyle::operator!= (const DisplayStyle & *other*) `[inline]`

Parameters:

other (fgColor!=other.fgColor) || (fgStyle!=other.fgStyle) || (bgColor!=other.bgColor)

6.15.3.9 Scr::DisplayStyle& Scr::DisplayStyle::operator= (const DisplayStyle & *other*) `[inline]`

Parameters:

other style, whose content will be assigned to this

Copy-assignment operator

6.15.4 Member Data Documentation

6.15.4.1 Uint32 Scr::DisplayStyle::style

As single unsigned integer - for easy copying

6.15.4.2 struct { ... } Scr::DisplayStyle::properties

And as a set of three separate variables, for easy manipulation

6.15.4.3 union { ... } `[private]`

style described as an union

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

- include/rexio/screen.h++
- lib/screen/src/core/displaystyle.c++

6.16 Scr::Exception Class Reference

base class for exceptions thrown by library objects.

`#include <commons.h++>`

Inherits std::exception.

Public Member Functions

- Exception (std::string _m) throw ()
- Exception (const Exception &_base) throw ()
- virtual const char ∗ what () const throw ()
- virtual ~Exception () throw ()

Private Attributes

- std::tr1::shared_ptr< std::string > message

6.16.1 Detailed Description

base class for exceptions thrown by library objects.

exception holds message about conditions etc, where it was thrown

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Exception::Exception (std::string *_m*) throw ()

Parameters:

_m message associated w/ exception. i.e. brief description of situation. Will be displayed after program failure.

Only argument is exception reason.

6.16.2.2 Exception::Exception (const Exception & *_base*) throw ()

Parameters:

_base exception to copy (copy constructor is used widely during throw-catch sequence.

6.16.2.3 Exception::∼Exception () throw () `[virtual]`

destructor conditionally frees resources (thanks to smart pointer used).

6.16.3 Member Function Documentation

6.16.3.1 const char ∗ Exception::what () const throw () `[virtual]`

what() derrivated from std::exception: informs on reason of exception

6.16.4 Member Data Documentation

6.16.4.1 std::tr1::shared_ptr<std::string> Scr::Exception::message `[private]`

message passed as reference counting pointer to prevent resource waste during throw-catch sequence.

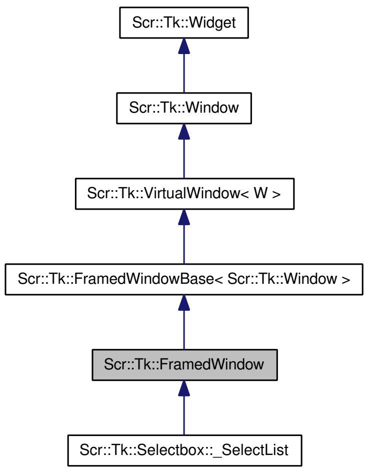
The documentation for this class was generated from the following files:

- include/rexio/commons.h++
- lib/screen/src/core/exception.c++

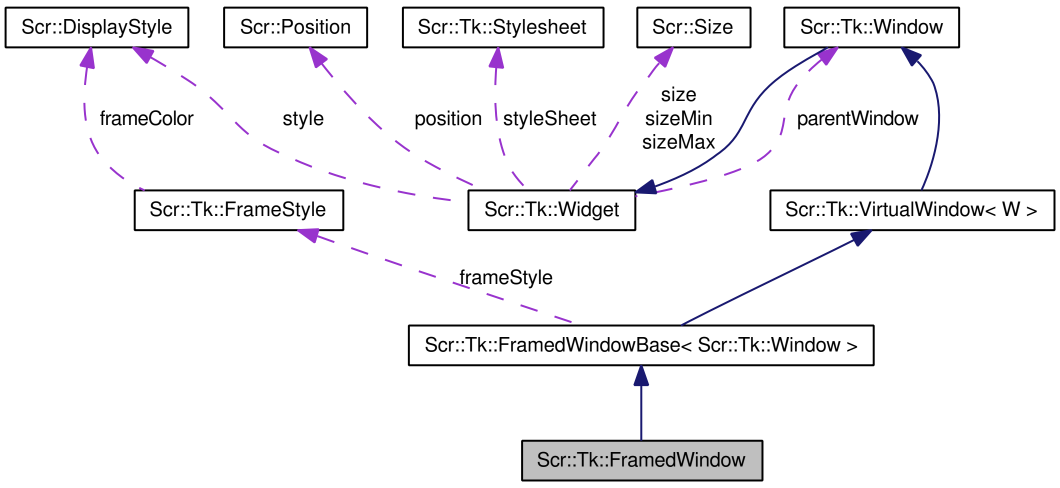
6.17 Scr::Tk::FramedWindow Class Reference

```
#include <framedwindow.h++>
```

Inheritance diagram for Scr::Tk::FramedWindow:


Scr::Tk::Widget
Scr::Tk::Window
Scr::Tk::VirtualWindow< W >
Scr::Tk::FramedWindowBase< Scr::Tk::Window >
Scr::Tk::FramedWindow
Scr::Tk::Selectbox::_SelectList

Collaboration diagram for Scr::Tk::FramedWindow:



Public Member Functions

- FramedWindow (Uint _height, Uint _width, const DisplayStyle &_style=FRAMEDWINDOW_-DEFAULT_STYLE, const FrameStyle &_frameStyle=FRAMEDWINDOW_DEFAULT_-FRAMESTYLE) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

6.17.1 Detailed Description

Basic FramedWindow with basic Window as its internal area.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 FramedWindow::FramedWindow (Uint *_height*, Uint *_width*, const DisplayStyle & *_-style* = `FRAMEDWINDOW_DEFAULT_STYLE`, const FrameStyle & *_frameStyle* = `FRAMEDWINDOW_-DEFAULT_FRAMESTYLE`) throw ()

Parameters:

_height desired height

_width desired width

_style optional style

_frameStyle optional frame style

6.17.3 Member Function Documentation

6.17.3.1 virtual bool Scr::Tk::FramedWindow::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::VirtualWindow< W >.

6.17.3.2 virtual const char* Scr::Tk::FramedWindow::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::VirtualWindow< W >.

6.17.3.3 virtual const char* Scr::Tk::FramedWindow::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::VirtualWindow< W >.

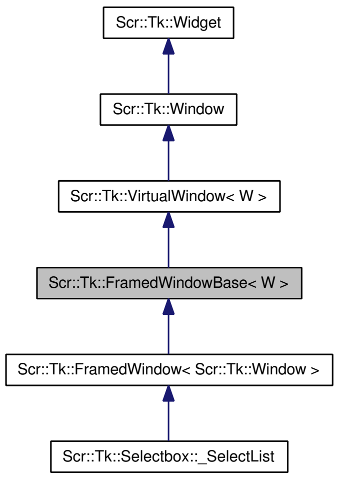
The documentation for this class was generated from the following files:

- include/rexio/tk/framedwindow.h++
- lib/toolkit/src/framedwindow.c++

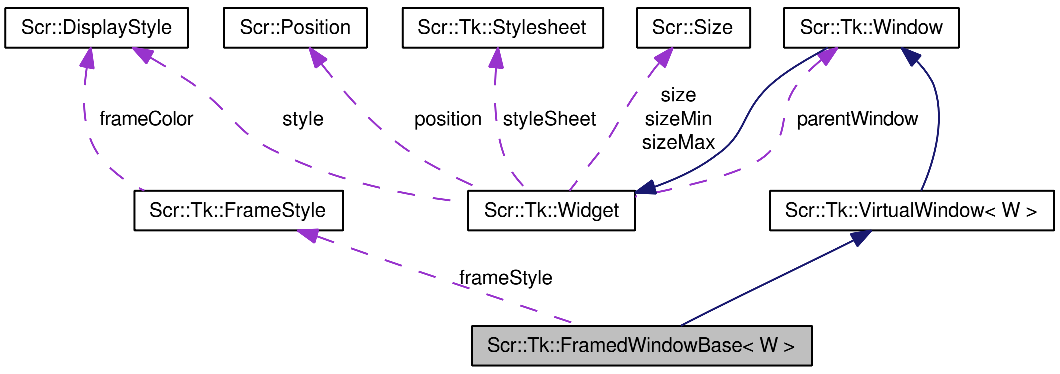
6.18 Scr::Tk::FramedWindowBase< W > Class Template Reference

`#include <framedwindow.h++>`

Inheritance diagram for Scr::Tk::FramedWindowBase< W >:


Scr::Tk::Widget
Scr::Tk::Window
Scr::Tk::VirtualWindow< W >
Scr::Tk::FramedWindowBase< W >
Scr::Tk::FramedWindow< Scr::Tk::Window >
Scr::Tk::Selectbox::_SelectList

Collaboration diagram for Scr::Tk::FramedWindowBase< W >:



Public Member Functions

- FramedWindowBase (Uint _height, Uint _width, const DisplayStyle &_style=FRAMEDWINDOW_DEFAULT_STYLE, const FrameStyle &_frameStyle=FRAMEDWINDOW_DEFAULT_FRAMESTYLE) throw ()
- virtual void OnResize () throw ()
- virtual void SetStylesheet (Stylesheet ∗_styleSheet) throw ()
- virtual void OnRedraw (Screen &screen) throw ()

Protected Attributes

- FrameStyle frameStyle

  how to draw a frame around inside.

6.18.1 Detailed Description

template<class W> class Scr::Tk::FramedWindowBase< W >

Parameters:

W class of inside's window. Template for all framed windows. FramedWindowBase is basically a window having a separate internal window to which most of the calls(like AddWidget) are routed.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 template<class W> Scr::Tk::FramedWindowBase< W >::FramedWindowBase (Uint *_height*, Uint *_width*, const DisplayStyle & *_style* = `FRAMEDWINDOW_DEFAULT_STYLE`, const FrameStyle & *_frameStyle* = `FRAMEDWINDOW_DEFAULT_FRAMESTYLE`) throw () `[inline]`

Parameters:

_height desired height

_width desired width

_style optional style

_frameStyle optional frame style

6.18.3 Member Function Documentation

6.18.3.1 template<class W> virtual void Scr::Tk::FramedWindowBase< W >::OnResize () throw () `[inline, virtual]`

Resize event. Do something i.e. adjust content to the new size. *VirtualWindow* specific: Has to be overloaded in deriving classes to handle proper resizing of containing window.

Implements Scr::Tk::VirtualWindow< W >.

Reimplemented in Scr::Tk::Selectbox::_SelectList.

6.18.3.2 template<class W> virtual void Scr::Tk::FramedWindowBase< W >::SetStylesheet (Stylesheet ∗ *_styleSheet*) throw () `[inline, virtual]`

Parameters:

_styleSheet pointer to style data

Apply Stylesheet to this widget. Reinitialize any style properties if their alternatives are supplied. *Window* specific: Recursively passes this call to all its children.

Reimplemented from Scr::Tk::Window.

6.18.3.3 template<class W> virtual void Scr::Tk::FramedWindowBase< W >::OnRedraw (Screen & *screen*) throw () `[inline, virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::VirtualWindow< W >.

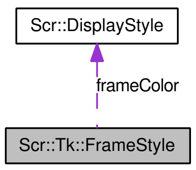
The documentation for this class was generated from the following file:

- include/rexio/tk/framedwindow.h++

6.19 Scr::Tk::FrameStyle Struct Reference

`#include <framedwindow.h++>`

Collaboration diagram for Scr::Tk::FrameStyle:



Public Member Functions

- FrameStyle (const DisplayStyle &_frameColor, wchar_t top=_DEFAULT_FRAME_TOP, wchar_t bottom=_DEFAULT_FRAME_BOTTOM, wchar_t left=_DEFAULT_FRAME_LEFT, wchar_t right=_DEFAULT_FRAME_RIGHT, wchar_t topLeft=_DEFAULT_FRAME_TOPLEFT, wchar_t topRight=_DEFAULT_FRAME_TOPRIGHT, wchar_t bottomLeft=_DEFAULT_FRAME_-BOTTOMLEFT, wchar_t bottomRight=_DEFAULT_FRAME_BOTTOMRIGHT)

Public Attributes

- DisplayStyle frameColor

  color of the frame

- union {
  };

  holds characters used for frame drawing

6.19.1 Detailed Description

Frame specific style.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Scr::Tk::FrameStyle::FrameStyle (const DisplayStyle & *_frameColor*, wchar_t *top* = `_DEFAULT_FRAME_TOP`, wchar_t *bottom* = `_DEFAULT_FRAME_BOTTOM`, wchar_t *left* = `_DEFAULT_FRAME_LEFT`, wchar_t *right* = `_DEFAULT_FRAME_RIGHT`, wchar_t *topLeft* = `_DEFAULT_FRAME_TOPLEFT`, wchar_t *topRight* = `_DEFAULT_FRAME_TOPRIGHT`, wchar_t *bottomLeft* = `_DEFAULT_FRAME_BOTTOMLEFT`, wchar_t *bottomRight* = `_DEFAULT_FRAME_BOTTOMRIGHT`) `[inline]`

Parameters:

_frameColor frame color

top

bottom

left

right

topLeft

topRight

bottomLeft

bottomRight

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

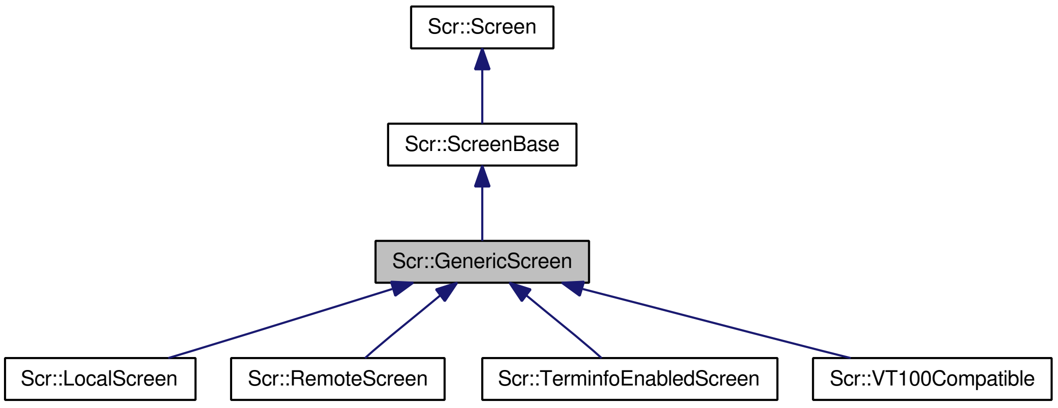
- include/rexio/tk/framedwindow.h++

6.20 Scr::GenericScreen Class Reference

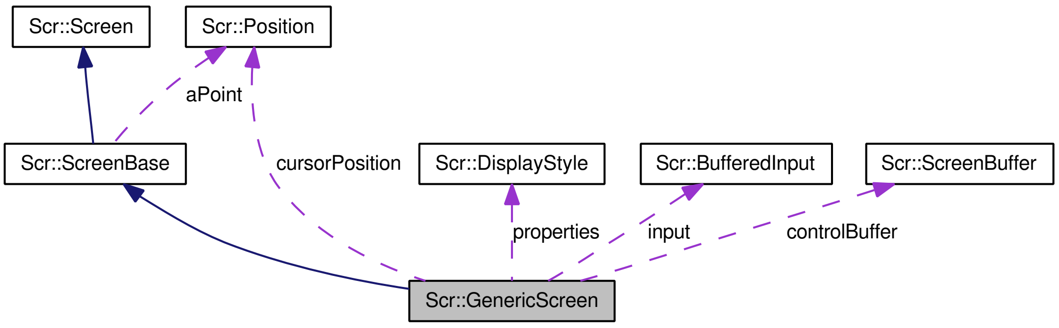
Most basic implementation of whole Scr::Screen.

`#include <genericscreen.h++>`

Inheritance diagram for Scr::GenericScreen:



Collaboration diagram for Scr::GenericScreen:



Public Member Functions

- GenericScreen (std::istream &_input, std::ostream &_output) throw ()
- virtual void Clear () throw ()
- virtual void SetBgColor (Bg::Color col) throw ()
- virtual void SetFgColor (Fg::Color col) throw ()
- virtual void SetFgStyle (Fg::Style s) throw ()
- virtual void GotoYX (Uint y, Uint x) throw (GotoOutOfRange)
- virtual void AddCharacter (char c) throw (PrintOutOfRange)
- virtual void AddCharacter (wchar_t c) throw (PrintOutOfRange, IllegalCharacter)
- virtual void ForceCursorPosition (Position p) throw (RangeError)
- virtual void AddText (const char ∗text) throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const std::string &text) throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const char ∗text, Uint cols, const std::vector< char > &widths) throw (PrintOutOfRange, IllegalCharacter)

- virtual void AddText (const std::wstring &text) throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const wchar_t ∗text) throw (PrintOutOfRange, IllegalCharacter)
- virtual Uint AddTextCols (const wchar_t ∗text, Uint limitcols) throw (PrintOutOfRange, IllegalCharacter)
- virtual Uint AddTextCols (const std::wstring &text, Uint limitcols) throw (PrintOutOfRange, IllegalCharacter)
- void AddSubscreenText (const char ∗text, Uint widthlimit) throw (PrintOutOfRange, IllegalCharacter)
- void AddSubscreenText (const wchar_t ∗text, Uint widthlimit) throw (PrintOutOfRange, IllegalCharacter)
- virtual void HorizontalLine (char c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void HorizontalLine (wchar_t c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void VerticalLine (char c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void VerticalLine (wchar_t c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void Rectangle (char c, const Size &s) throw (PrintOutOfRange, IllegalCharacter)
- virtual void Rectangle (wchar_t c, const Size &s) throw (PrintOutOfRange, IllegalCharacter)
- virtual void HideCursor () throw (CursorVisibilityNotSupported)
- virtual void ShowCursor () throw (CursorVisibilityNotSupported)
- void Refresh () throw (ConnectionError)
- virtual Screen ∗ CreateSubScreen (Uint _y_offset, Uint _x_offset, Uint _h, Uint _w) throw (SubscreenOutOfRange)
- virtual const char ∗ GetType () const throw (TerminalTypeUnknown)
- virtual Uint GetHeight () const throw ()
- virtual Uint GetWidth () const throw ()
- virtual bool GetCursorVisibility () const throw ()
- virtual void CleanUp () throw (ConnectionError)
- virtual void Resize (Uint rows, Uint cols) throw ()
- template<>
  Uint PrecomputeTextCharsWidth (const char ∗text, vector< char > &widths, Uint maxwidth) throw (RangeError, IllegalCharacter)

  local template specialization: adds UTF8 Decoding

Protected Member Functions

- virtual Key DecodeKeyPressed () throw (Connection::UnsupportedKey,Screen::InvalidUTF8)

Protected Attributes

- ScreenBuffer controlBuffer
- DisplayStyle properties
- Position cursorPosition
- std::ostream & output

Private Member Functions

- template<typename _char_type>
  Uint PrecomputeTextCharsWidth (_char_type ∗text, std::vector< char > &widths, Uint maxwidth) throw (RangeError, IllegalCharacter)

6.20.1 Detailed Description

Most basic implementation of whole Scr::Screen.

This class provides generic implementation of large part of Scr::Screen interface, including basic output subroutines, but some of them lacks important platform-specific features

6.20.2 Constructor & Destructor Documentation

6.20.2.1 GenericScreen::GenericScreen (std::istream & *_input*, std::ostream & *_output*) throw ()

Parameters:

_input

_output GenericCcreen operates on C++ standard iostream.

6.20.3 Member Function Documentation

6.20.3.1 template<typename _char_type> Uint GenericScreen::PrecomputeTextCharsWidth (_-char_type ∗ *text*, std::vector< char > & *widths*, Uint *maxwidth*) throw (RangeError, IllegalCharacter) `[inline, private]`

Function used to compute width of text as well as width of each character. The function is designed to be called from within all types of AddText

Returns:

width of string (correct value <= maxwidth)

Parameters:

text is text, whose element widths need to be computed

widths is C-type array of character widths, that need to be computed

maxwidth is max width of whole text (if width of whole text exceeds allowed width, stop computation and throw exception)

Exceptions:

Scr::Screen::RangeError exception is thrown when text is too wide.

Scr::Screen::IllegalCharacter exception is thrown when UNICODE encoding is incorrect (validation occurs only for _char_type=char)

6.20.3.2 Scr::Key Scr::GenericScreen::DecodeKeyPressed () throw (Connection::UnsupportedKey,Screen::InvalidUTF8) `[protected, virtual]`

get key esc-code from std input stream. decode it into form from keyboard.h++

Reimplemented in Scr::TerminfoEnabledScreen, and Scr::VT100Compatible.

6.20.3.3 void GenericScreen::Clear () throw () `[virtual]`

empty controlBuffer

Implements Scr::Screen.

6.20.3.4 void GenericScreen::SetBgColor (Bg::Color *col*) throw () `[virtual]`

Parameters:

col new background colour to be set

Returns:

nothing upon successful execution

Function operates on properties member object.

Refer to manual for base class for action description.

Implements Scr::Screen.

6.20.3.5 void GenericScreen::SetFgColor (Fg::Color *col*) throw () `[virtual]`

Parameters:

col new foreground colour to be set

Returns:

nothing upon successful execution

Function operates on properties member object.

Refer to manual for base class for action description.

Implements Scr::Screen.

6.20.3.6 void GenericScreen::SetFgStyle (Fg::Style *s*) throw () `[virtual]`

Parameters:

s new foreground text style to be set

Returns:

nothing upon successful execution

Function operates on properties member object.

Refer to manual for base class for action description.

Implements Scr::Screen.

6.20.3.7 void GenericScreen::GotoYX (Uint *y*, Uint *x*) throw (GotoOutOfRange) `[virtual]`

Parameters:

y

x new coordinates of active point (please remember the order of theese attributes)

Operates on coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.8 void GenericScreen::AddCharacter (char *c*) throw (PrintOutOfRange) `[virtual]`

Parameters:

c

Operates on controlBuffer and coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.9 void GenericScreen::AddCharacter (wchar_t *c*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c

Operates on controlBuffer and coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.10 void GenericScreen::ForceCursorPosition (Position *p*) throw (RangeError) `[virtual]`

Parameters:

p position

visible after refresh

Implements Scr::Screen.

6.20.3.11 void GenericScreen::AddText (const char ∗ *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text text to be printed (as C string)

Note:

Operates on controlBuffer and coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.12 void GenericScreen::AddText (const std::string & *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text what to be printed (as C++ string)

Note:

Operates on controlBuffer and coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.13 void GenericScreen::AddText (const char $*$ *text*, Uint *cols*, const std::vector< char > & *widths*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text UTF-8 encoded character string

cols length of string

widths widths of subsequent characters

Function prints specified text assuming, that its width is EXACTLY specified by cols parameter

Exceptions:

PrintOutOfRange is thrown if initial position of active point is invalid, or if text is too long (as function does not support line breaks).

IllegalCharacter will be thrown if text supplied is not a valid UTF-8 string (even "overlong sequences" will be considered illegal (according to an apropriate RFC

Note:

function is NOT a part of Scr::Screen interface, and is not accessible outside of screen module

See also:

Screen::AddText(const char $*$ text) for extensive description

6.20.3.14 void GenericScreen::AddText (const std::wstring & *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text

Operates on controlBuffer and coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.15 void GenericScreen::AddText (const wchar_t $*$ *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text

Operates on controlBuffer and coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.16 Uint GenericScreen::AddTextCols (const wchar_t $*$ *text*, Uint *limitcols*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text wide string

limitcols max width in columns

Function prints AT MOST limitcols wide string. Width means number of columns, which is not the same thing as number of characters, as most CJK glyphs are multicolumn.

Exceptions:

PrintOutOfRange is thrown if initial position of active point is invalid, or if text is too long (as function does not support line breaks).

IllegalCharacter will be thrown if text supplied is not a valid UTF-8 string (even "overlong sequences" will be considered illegal (according to an apropriate RFC

See also:

Screen::AddText(const char ∗ text) for extensive description

Note:

Operates on controlBuffer and coordinate values inherited from ScreenBase.

Implements Scr::Screen.

6.20.3.17 Uint GenericScreen::AddTextCols (const std::wstring & *text*, Uint *limitcols*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text wide string

limitcols max width in columns

Function prints AT MOST limitcols wide string. Width means number of columns, which is not the same thing as number of characters, as most CJK glyphs are multicolumn.

Exceptions:

PrintOutOfRange is thrown if initial position of active point is invalid, or if text is too long (as function does not support line breaks).

IllegalCharacter will be thrown if text supplied is not a valid UTF-8 string (even "overlong sequences" will be considered illegal (according to an apropriate RFC

See also:

Screen::AddText(const char ∗ text) for extensive description

Operates on controlBuffer and coordinate values inherited from ScreenBase

Implements Scr::Screen.

6.20.3.18 void GenericScreen::AddSubscreenText (const char ∗ *text*, Uint *widthlimit*) throw (PrintOutOfRange, IllegalCharacter)

Function adds "text in subscreen", that is text, which was to be be inserted in subscreen. This function is called by apropriate Scr::Subscreen::AddText .

Parameters:

text UTF-8 encoded text to be printed

width maximum number of columns to be printed

Exceptions:

Scr::Screen::IllegalCharacter may be thrown if any character of text is incorrectly encoded

Scr::Screen::PrintOutOfRange is thrown when text runs out of root screen range or when it's width (as number of columns, not characters) exceeds widthlimit.

6.20.3.19 void GenericScreen::AddSubscreenText (const wchar_t * *text*, Uint *widthlimit*) throw (PrintOutOfRange, IllegalCharacter)

Purpose of this function is as above, but one of parameters slightly differs.

Parameters:

text UNICODE text

width maximum number of columns to be printed

Exceptions:

Scr::Screen::PrintOutOfRange is thrown when text runs out of root screen range or when it's width (as number of columns, not characters) exceeds widthlimit. *

6.20.3.20 void GenericScreen::HorizontalLine (char *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c ASCII character

n number of repetitions (length of line)

Function adds horizontal line of n characters c.

Implements Scr::Screen.

6.20.3.21 void GenericScreen::HorizontalLine (wchar_t *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c UNICODE character

n number of repetitions (length of line)

Function adds horizontal line of n characters c.

Implements Scr::Screen.

6.20.3.22 void GenericScreen::VerticalLine (char *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

- ***c*** ASCII character
- ***n*** number of repetitions (length of line)

Function adds verticel line of n characters c.

Implements Scr::Screen.

6.20.3.23 void GenericScreen::VerticalLine (wchar_t *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

- ***c*** UNICODE character
- ***n*** number of repetitions (length of line)

Function adds vertical line of n characters c.

Implements Scr::Screen.

6.20.3.24 void GenericScreen::Rectangle (char *c*, const Size & *s*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

- ***c*** character used to create rectangle
- ***s*** dimensions of rectangle

Function creates rectangle of characters. s specifies number of rows and number of repetitions of character c in each row.

Implements Scr::Screen.

6.20.3.25 void GenericScreen::Rectangle (wchar_t *c*, const Size & *s*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

- ***c*** character used to create rectangle
- ***s*** dimensions of rectangle

Function creates rectangle of characters. s specifies number of rows and number of repetitions of character c in each row.

Implements Scr::Screen.

6.20.3.26 void GenericScreen::HideCursor () throw (CursorVisibilityNotSupported) `[virtual]`

make cursor invisible

Implements Scr::Screen.

6.20.3.27 void GenericScreen::ShowCursor () throw (CursorVisibilityNotSupported) `[virtual]`

make it visible again

Implements Scr::Screen.

6.20.3.28 void GenericScreen::Refresh () throw (ConnectionError) `[virtual]`

Most basic implementation suitable really only for dumb terminals or line printers: prints each line of buffer to stdout. Created only for debugging reasons.

Implements Scr::Screen.

Reimplemented in Scr::TerminfoEnabledScreen, and Scr::VT100Compatible.

6.20.3.29 Screen ∗ GenericScreen::CreateSubScreen (Uint *_y_offset*, Uint *_x_offset*, Uint *_h*, Uint *_w*) throw (SubscreenOutOfRange) `[virtual]`

Parameters:

_y_offset vertical offset from top edge of this screen to top edge of new SubScreen.

_x_offset horizontal offser

_h height

_w with

Returns:

pointer to new SubScreen (programmer will have to free it's resources to prevent memory leak and other errors).

Exceptions:

Scr::Screen::SubscreenOutOfRange is thrown when too big subscreen requested or inapropriate position specified

Implements Scr::Screen.

6.20.3.30 const char ∗ Scr::GenericScreen::GetType () const throw (TerminalTypeUnknown) `[virtual]`

Returns:

always throw exceptn

Implements Scr::Screen.

Reimplemented in Scr::LocalScreen, and Scr::RemoteScreen.

6.20.3.31 Uint GenericScreen::GetHeight () const throw () `[virtual]`

Returns:

height of controlBuffer

Implements Scr::Screen.

6.20.3.32 Uint GenericScreen::GetWidth () const throw () `[virtual]`

Returns:

width of controlBuffer

Implements Scr::Screen.

6.20.3.33 bool GenericScreen::GetCursorVisibility () const throw () `[virtual]`

Returns:

true if cursor is visible, false if it ishidden

See also:

ShowCursor HideCursor

Implements Scr::Screen.

6.20.3.34 void GenericScreen::CleanUp () throw (ConnectionError) `[virtual]`

Cleans screen up: restore default colours and clear (it is good to use this function while finishing application etc.)

Reimplemented in Scr::TerminfoEnabledScreen, and Scr::VT100Compatible.

6.20.3.35 void GenericScreen::Resize (Uint *rows*, Uint *cols*) throw () `[virtual]`

Parameters:

rows new number of rows (new height) of screen

cols new number of columns of screen

Returns:

nothing upon successful execution

Change the output size.

Note:

this function does not change size of physical screen, it may only be used to resize this object to fit physical screen size. If screen grows, new characters are filled with current background colour. Function does not refresh the physical screen after it's resizing, so it's content is undefined after call of this function (however left-top part of it will be restored after Refresh call).

Exceptions:

Scr::Screen::Exception::IllegalOperation if particular screen may nor be resized for some implementation- or platform- specific reasons. In particular case primitive subscreens may not be resized (SubscreenResize specialization of exception is thrown then).

Implements Scr::Screen.

Reimplemented in Scr::TerminfoEnabledScreen, and Scr::VT100Compatible.

6.20.4 Member Data Documentation

6.20.4.1 ScreenBuffer Scr::GenericScreen::controlBuffer `[protected]`

buffer used to implement all textual operations. All Add∗ functions operate on it directly.

6.20.4.2 DisplayStyle Scr::GenericScreen::properties `[protected]`

current properties (set w/ SetBg/FgColor/Style)

6.20.4.3 Position Scr::GenericScreen::cursorPosition `[protected]`

cursorPosition

6.20.4.4 std::ostream& Scr::GenericScreen::output `[protected]`

Output file stream for writing

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

- lib/screen/include/genericscreen.h++
- lib/screen/src/real/genericscreen.c++

6.21 Scr::GlyphWidth Class Reference

`#include <glyphwidth.h++>`

Static Public Member Functions

- static Uint Get (wchar_t ch)

Static Private Attributes

- static std::bitset<(1<< 17)∗2 > glyphWidth

6.21.1 Detailed Description

Singleton class.

6.21.2 Member Function Documentation

6.21.2.1 static Uint Scr::GlyphWidth::Get (wchar_t *ch*) `[inline, static]`

- ch

Returns:

width of unicode character (0 or 1 or 2), that means number of cells in console, it needs to fit.

6.21.3 Member Data Documentation

6.21.3.1 std::bitset<(1<< 17)∗2 > Scr::GlyphWidth::glyphWidth `[static, private]`

Bitset for caching the width results. 2 bits per glyph. Note: the bitset gets reasonably fast only in the Release build

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

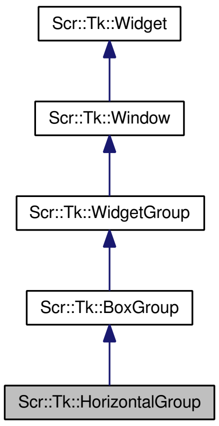
- include/rexio/glyphwidth.h++
- lib/screen/src/core/glyphwidth.c++

6.22 Scr::Tk::HorizontalGroup Class Reference

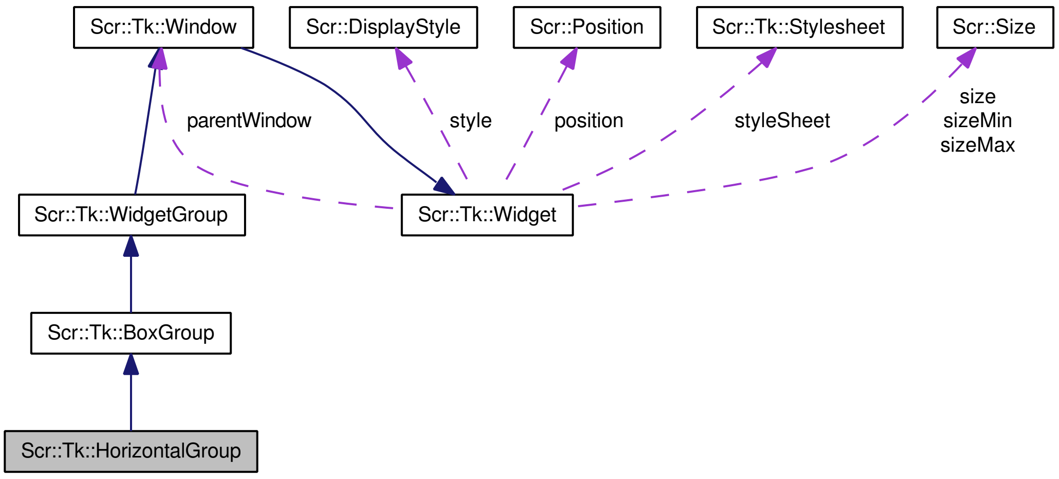
Horizontal widget grouping capabilities.

`#include <horizontalgroup.h++>`

Inheritance diagram for Scr::Tk::HorizontalGroup:



Collaboration diagram for Scr::Tk::HorizontalGroup:



Public Member Functions

- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Member Functions

- virtual void ArrangeContents () throw ()

6.22.1 Detailed Description

Horizontal widget grouping capabilities.

Intelligently places the containing widgets among allocated space. Widgets are placed horizontally.

6.22.2 Member Function Documentation

6.22.2.1 void HorizontalGroup::ArrangeContents () throw () `[protected, virtual]`

where all magic is done :)

Implements Scr::Tk::BoxGroup.

6.22.2.2 virtual bool Scr::Tk::HorizontalGroup::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::BoxGroup.

6.22.2.3 virtual const char∗ Scr::Tk::HorizontalGroup::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::BoxGroup.

6.22.2.4 virtual const char∗ Scr::Tk::HorizontalGroup::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::BoxGroup.

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

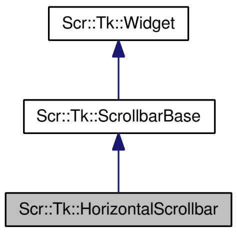
- include/rexio/tk/horizontalgroup.h++
- lib/toolkit/src/horizontalgroup.c++

6.23 Scr::Tk::HorizontalScrollbar Class Reference

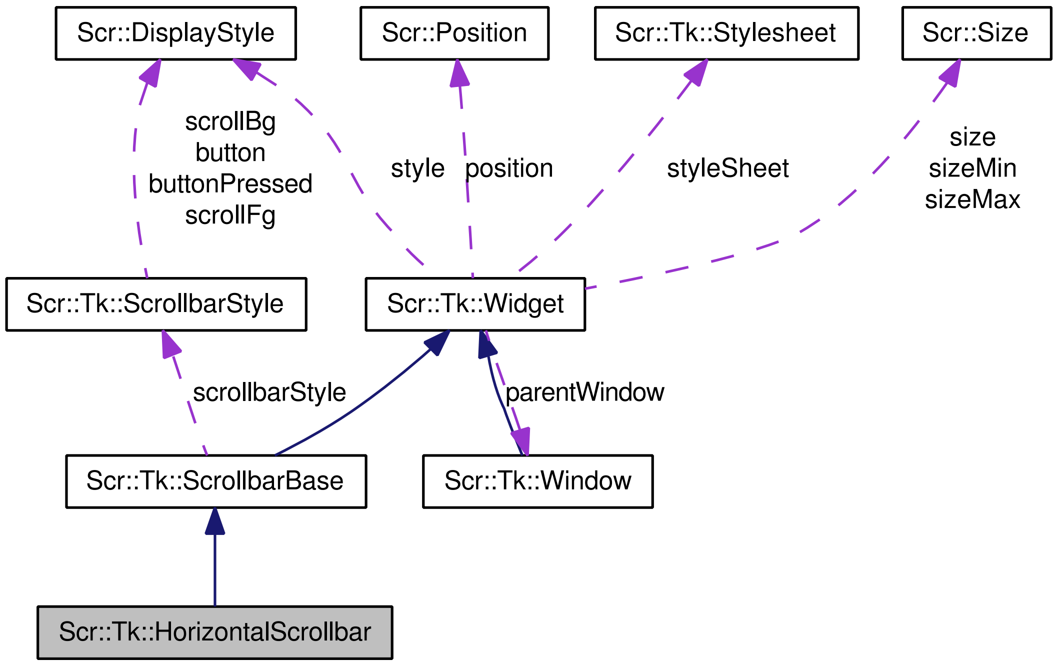
Horizontal scrollbar.

`#include <scrollbar.h++>`

Inheritance diagram for Scr::Tk::HorizontalScrollbar:



Collaboration diagram for Scr::Tk::HorizontalScrollbar:



Public Member Functions

- HorizontalScrollbar (Uint _width, const ScrollbarStyle &_scrollbarStyle=ScrollbarStyle()) throw ()
- virtual void OnRedraw (Screen &screen) throw ()

- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

6.23.1 Detailed Description

Horizontal scrollbar.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 HorizontalScrollbar::HorizontalScrollbar (Uint *_width*, const ScrollbarStyle & *_scrollbarStyle* = ScrollbarStyle`()`) throw ()

Parameters:

_width

_scrollbarStyle

6.23.3 Member Function Documentation

6.23.3.1 void HorizontalScrollbar::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Implements Scr::Tk::ScrollbarBase.

6.23.3.2 virtual bool Scr::Tk::HorizontalScrollbar::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::ScrollbarBase.

6.23.3.3 virtual const char∗ Scr::Tk::HorizontalScrollbar::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::ScrollbarBase.

6.23.3.4 virtual const char∗ Scr::Tk::HorizontalScrollbar::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::ScrollbarBase.

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

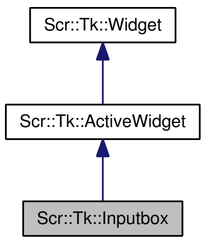
- include/rexio/tk/scrollbar.h++
- lib/toolkit/src/scrollbar.c++

6.24 Scr::Tk::Inputbox Class Reference

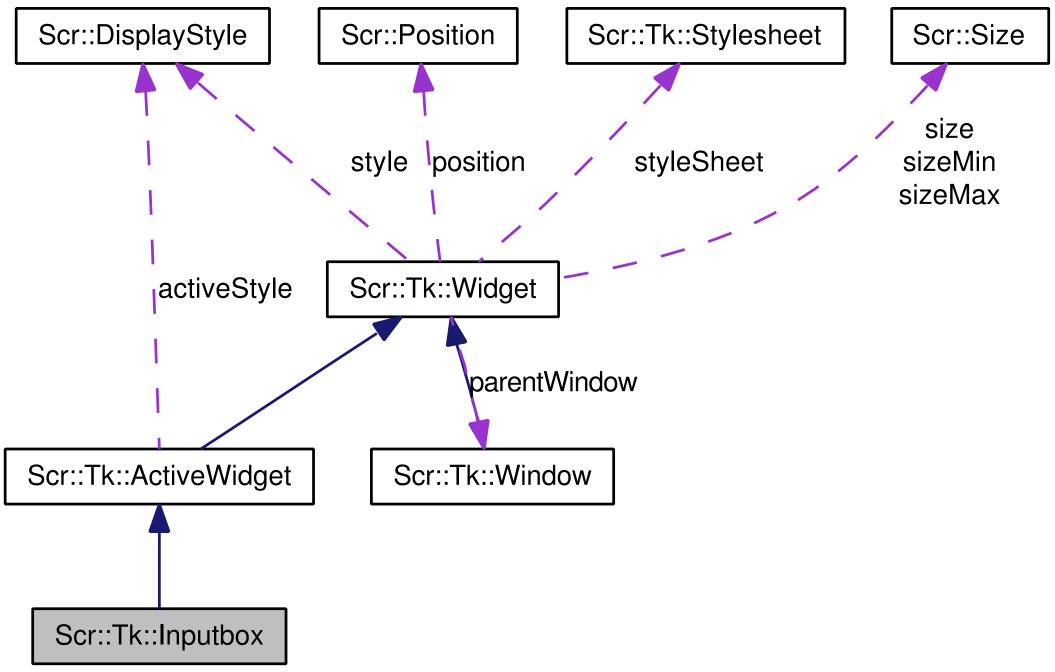
Simple text input field.

`#include <inputbox.h++>`

Inheritance diagram for Scr::Tk::Inputbox:



Collaboration diagram for Scr::Tk::Inputbox:



Public Member Functions

- virtual void SetText (const std::wstring &_text) throw ()
- virtual const std::wstring & GetText () throw ()
- virtual void SetMaxLength (Uint _maxLength) throw ()
- virtual Uint GetMaxLength () throw ()
- virtual void SetOffset (Uint _textOffset) throw (OffsetOutOfRange)
- virtual Uint GetOffset () throw ()
- virtual void SetStylesheet (Stylesheet ∗_styleSheet) throw ()
- virtual void OnKeyDown (Key key) throw ()
- virtual void OnRedraw (Screen &screen) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Attributes

- Uint textOffset

  Index of first character currently visible in the input.

- std::wstring text

  Text content..

- InputboxStyle inputboxStyle

  Inputbox specific style.

- Uint maxLength

  Maximum length of input.

Private Attributes

- Uint cursorPos
- Uint charPos
- Uint curCols

  Currently shown number of columns.

- Uint curChars

  Currently shown number of characters.

6.24.1 Detailed Description

Simple text input field.

6.24.2 Member Function Documentation

6.24.2.1 void Inputbox::SetText (const std::wstring & *_text*) throw () `[virtual]`

Parameters:

_text Extended string to replace current content of inputbox

Set the actual content text.

6.24.2.2 const std::wstring & Inputbox::GetText () throw () `[virtual]`

Returns:

const reference to the containing text

Get the content text.

6.24.2.3 void Inputbox::SetMaxLength (Uint *_maxLength*) throw () `[virtual]`

Parameters:

_maxLength new value

Set max length of possible input

6.24.2.4 Uint Inputbox::GetMaxLength () throw () `[virtual]`

Returns:

maxLength

Get max length of possible input

6.24.2.5 void Inputbox::SetOffset (Uint *_textOffset*) throw (OffsetOutOfRange) `[virtual]`

Parameters:

_textOffset new value

Set new text offset.

Exceptions:

OffsetOutOfRange is thrown had the offset been wrongly provided.

6.24.2.6 Uint Inputbox::GetOffset () throw () `[virtual]`

Returns:

textOffset

Return current text offset.

6.24.2.7 virtual void Scr::Tk::Inputbox::SetStylesheet (Stylesheet ∗ *_styleSheet*) throw () `[inline, virtual]`

Parameters:

_styleSheet pointer to style data

Apply Stylesheet to this widget. Reinitialize any style properties if their alternatives are supplied.

Reimplemented from Scr::Tk::ActiveWidget.

6.24.2.8 void Inputbox::OnKeyDown (Key *key*) throw () `[virtual]`

Parameters:

key keycode

Keyboard button press event.

Reimplemented from Scr::Tk::ActiveWidget.

6.24.2.9 void Inputbox::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Widget.

6.24.2.10 virtual bool Scr::Tk::Inputbox::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::ActiveWidget.

6.24.2.11 virtual const char∗ Scr::Tk::Inputbox::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::ActiveWidget.

6.24.2.12 virtual const char∗ Scr::Tk::Inputbox::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::ActiveWidget.

6.24.3 Member Data Documentation

6.24.3.1 Uint Scr::Tk::Inputbox::cursorPos `[private]`

Column position of the cursor. 0 is considered beginning of Inputbox

6.24.3.2 Uint Scr::Tk::Inputbox::charPos `[private]`

After which character in the current input the cursor is located

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

- include/rexio/tk/inputbox.h++
- lib/toolkit/src/inputbox.c++

6.25 Scr::Key Class Reference

Class represents key (or key combination) pressed on client terminal.

`#include <keyboard.h++>`

Public Types

- enum ASCII
- enum Special

Public Member Functions

- Key (Uint key) throw ()
- operator Uint () throw ()
- bool IsABasicKey () throw ()
- char GetBasicKey () throw (NotABasicKey)
- const char ∗ GetKeyName () throw ()

Static Private Attributes

- static const Uint specialMask = 0x56000000

  special key pressed

6.25.1 Detailed Description

Class represents key (or key combination) pressed on client terminal.

6.25.2 Member Enumeration Documentation

6.25.2.1 enum Scr::Key::ASCII

Special ascii keys as teletype mnemonics. Please note, that this enum is temporary, and will be deprecated in 1.1

6.25.2.2 enum Scr::Key::Special

Special key names. May be used for comparizons against key object (please refer to handbook for use example)

6.25.3 Constructor & Destructor Documentation

6.25.3.1 Scr::Key::Key (Uint *key*) throw () `[inline]`

Parameters:

key unsigned integer form

This constructor allows implicit conversion between two forms of key

6.25.4 Member Function Documentation

6.25.4.1 Scr::Key::operator Uint () throw () `[inline]`

implicit or static cast operator

6.25.4.2 bool Scr::Key::IsABasicKey () throw () `[inline]`

If represents plain ascii character, function returns true. false is returned otherwise

6.25.4.3 char Key::GetBasicKey () throw (NotABasicKey)

as if it was a letter A-Z

6.25.4.4 const char ∗ Key::GetKeyName () throw ()

KEYD(Up);

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

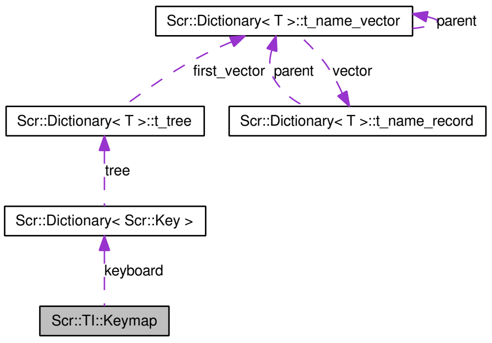
- include/rexio/keyboard.h++
- lib/screen/src/core/keyboard.c++

6.26 Scr::TI::Keymap Class Reference

Class responsible for mapping control sequences to unique key codes.

`#include <terminfokeymap.h++>`

Collaboration diagram for Scr::TI::Keymap:


Scr::Dictionary< T >::t_name_vector
parent
first_vector
parent
vector
Scr::Dictionary< T >::t_tree
Scr::Dictionary< T >::t_name_record
tree
Scr::Dictionary< Scr::Key >
keyboard
Scr::TI::Keymap

Public Member Functions

- validity TestCode (const char ∗code) throw ()

Protected Member Functions

- Keymap (const TerminfoEntry &te) throw ()
- virtual void InitializeKeymap (const TerminfoEntry &te) throw ()

Private Attributes

- key_dictionary keyboard

  real engine of this module is Dictionary Tree.

6.26.1 Detailed Description

Class responsible for mapping control sequences to unique key codes.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Keymap::Keymap (const TerminfoEntry & *te*) throw () `[explicit, protected]`

Parameters:

te Terminfo entry for which keymap will be generated

6.26.3 Member Function Documentation

6.26.3.1 void Keymap::InitializeKeymap (const TerminfoEntry & *te*) throw () `[protected, virtual]`

Do real work of constructor. Way of doing this work may slightly differ for specific terminal types, so we have to move this action from the constructor to enable virtualization

Parameters:

te Terminfo entry for which keymap will be generated

6.26.3.2 Keymap::validity Keymap::TestCode (const char ∗ *code*) throw ()

Parameters:

code keycode provided by client. i.e. "\x1b[24∼" means function key F12 for DEC VT220 Terminal.

Test if string supplied matches any key code stored in tree.

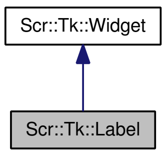
The documentation for this class was generated from the following files:

- lib/screen/include/terminfokeymap.h++
- lib/screen/src/terminfo/terminfokeymap.c++

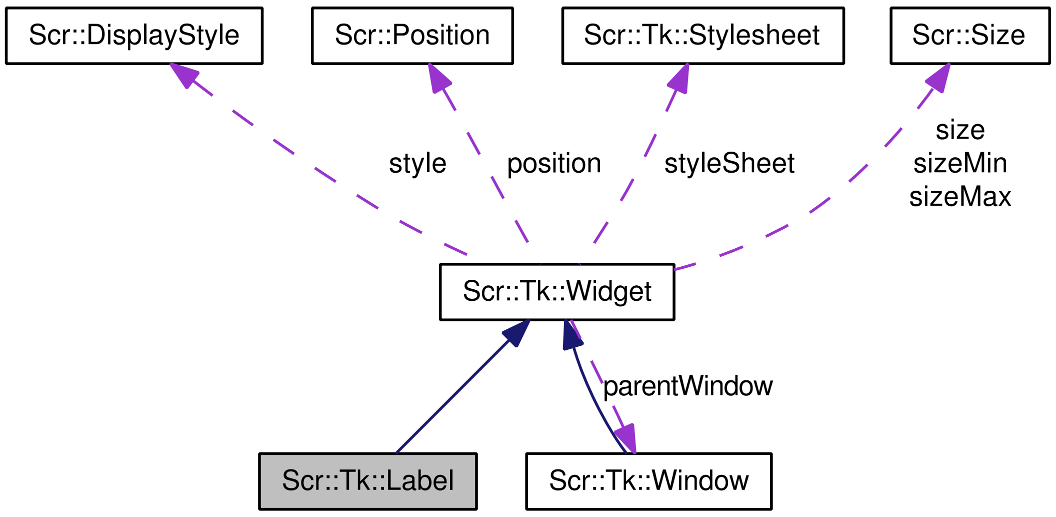
6.27 Scr::Tk::Label Class Reference

`#include <label.h++>`

Inheritance diagram for Scr::Tk::Label:



Collaboration diagram for Scr::Tk::Label:



Public Member Functions

- virtual void SetStylesheet (Stylesheet ∗_styleSheet) throw ()
- virtual const std::string & GetText () const throw ()
- virtual void SetText (const std::string _label) throw ()
- virtual void OnFocus (FocusPolicy focustype) throw ()
- virtual void OnUnFocus (FocusPolicy focustype) throw ()
- virtual void OnRedraw (Screen &screen) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Attributes

- std::string label

6.27.1 Detailed Description

Simple text data holder.

6.27.2 Member Function Documentation

6.27.2.1 virtual void Scr::Tk::Label::SetStylesheet (Stylesheet ∗ *_styleSheet*) throw () `[inline, virtual]`

Parameters:

_styleSheet pointer to style data

Apply Stylesheet to this widget. Reinitialize any style properties if their alternatives are supplied.

Reimplemented from Scr::Tk::Widget.

6.27.2.2 const std::string & Label::GetText () const throw () `[virtual]`

Returns:

containing text

Return the actual label text.

6.27.2.3 void Label::SetText (const std::string *_label*) throw () `[virtual]`

Parameters:

_label string to replace current content of label

Set the actual label text.

6.27.2.4 void Label::OnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element focused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Widget.

6.27.2.5 void Label::OnUnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element unfocused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Widget.

6.27.2.6 void Label::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Widget.

6.27.2.7 virtual bool Scr::Tk::Label::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::Widget.

6.27.2.8 virtual const char* Scr::Tk::Label::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::Widget.

6.27.2.9 virtual const char* Scr::Tk::Label::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::Widget.

6.27.3 Member Data Documentation

6.27.3.1 std::string Scr::Tk::Label::label `[protected]`

Actual label holder.

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

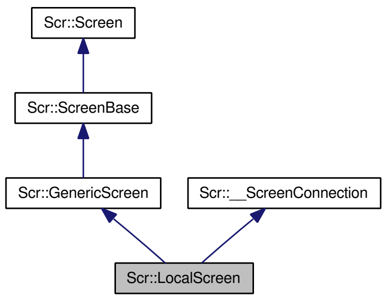
- include/rexio/tk/label.h++
- lib/toolkit/src/label.c++

6.28 Scr::LocalScreen Class Reference

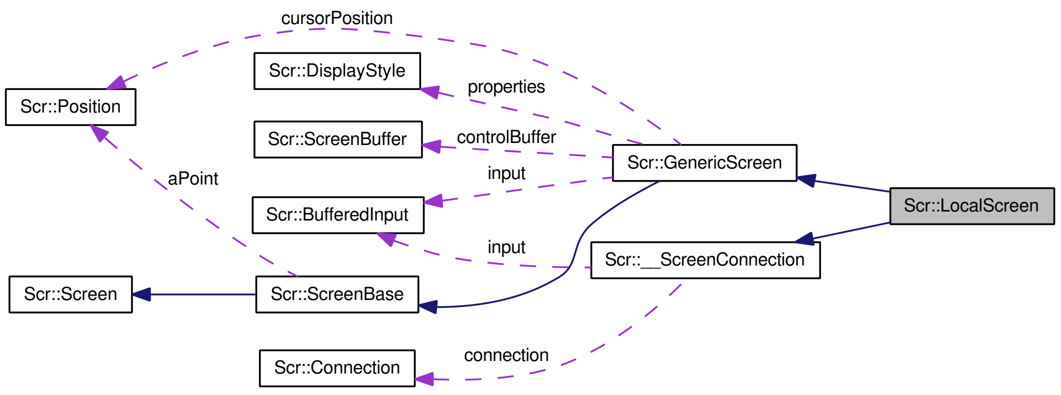
connection on localhost, using cin/cout

`#include <localscreen.h++>`

Inheritance diagram for Scr::LocalScreen:



Collaboration diagram for Scr::LocalScreen:



Public Member Functions

- void TestForResize ()
- LocalScreen (Connection &_connection, std::istream &_input, std::ostream &_output) throw ()
- virtual const char ∗ GetType () const throw ()
- int ProcessConnection ()

Private Attributes

- struct termios term

6.28.1 Detailed Description

connection on localhost, using cin/cout

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Scr::LocalScreen::LocalScreen (Connection & *_connection*, std::istream & *_input*, std::ostream & *_output*) throw ()

Parameters:

_connection reference to object representing connection itself

_input reference to input stream

_output reference to output stream

please note, that, turning sync. off for cin may be detected as memory leak by valgrind debugger. According to GNU folks thic behaviour is normal (since desynchronizing means allocating special memory block, which is never freed as standard streams are never deleted) `http://gcc.gnu.org/ml/gcc-bugs/2006-06/msg00824.html`

6.28.3 Member Function Documentation

6.28.3.1 void Scr::LocalScreen::TestForResize ()

Parameters:

infd file descriptor

Returns:

true if size changed

Function checks if size set for object equals size of appropriate screen. If it differs, Resize() is called to match changes

6.28.3.2 const char ∗ Scr::LocalScreen::GetType () const throw () `[virtual]`

Returns:

getenv("TERM");

Reimplemented from Scr::GenericScreen.

6.28.3.3 int Scr::LocalScreen::ProcessConnection () `[virtual]`

basic main loop of application using local screen

Implements Scr::__ScreenConnection.

6.28.4 Member Data Documentation

6.28.4.1 struct termios Scr::LocalScreen::term `[read, private]`

Store initial terminal settings to restore them after finishing connection (especially settings connected with local echo.

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

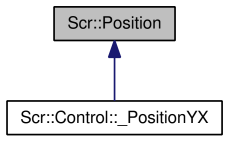
- lib/screen/include/localscreen.h++
- lib/screen/src/real/localscreen.c++

6.29 Scr::Position Struct Reference

position container.

`#include <commons.h++>`

Inheritance diagram for Scr::Position:



Public Member Functions

- Position (Uint _row, Uint _col)
- Position operator+ (const Position &pos)
- Position operator+ (const Size &size)
- Position operator+ (const Vector &vec)
- Position & operator+= (const Position &pos)
- Position & operator+= (const Size &size)
- Position & operator+= (const Vector &vec)
- Position operator- (const Position &pos)
- Position operator- (const Size &size)
- Position operator- (const Vector &vec)
- Position & operator-= (const Position &pos)
- Position & operator-= (const Size &size)
- Position & operator-= (const Vector &vec)

Public Attributes

- Uint row
- Uint col

6.29.1 Detailed Description

position container.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 Position::Position (Uint *_row*, Uint *_col*)

Parameters:

_row	row position
_col	col position

Simple constructor for convenient initialization and creation.

6.29.3 Member Function Documentation

6.29.3.1 Position Position::operator+ (const Position & *pos*)

Parameters:

pos addition target

Simple addition.

6.29.3.2 Position Position::operator+ (const Size & *size*)

Parameters:

size size to increment by

Result of incrementing position by a size of some object.

6.29.3.3 Position Position::operator+ (const Vector & *vec*)

Parameters:

vec vector to add

Result of modificating position by a vector.

6.29.3.4 Position & Position::operator+= (const Position & *pos*)

Parameters:

pos addition target

Simple assignment by addition.

6.29.3.5 Position & Position::operator+= (const Size & *size*)

Parameters:

size size to increment by

Incrementation of position by a size of some object.

6.29.3.6 Position & Position::operator+= (const Vector & *vec*)

Parameters:

vec vector to add

Modification of position by a vector.

6.29.3.7 Position Position::operator- (const Position & *pos*)

Parameters:

pos substraction target

Simple substraction.

6.29.3.8 Position Position::operator- (const Size & *size*)

Parameters:

size size to decrement by

Result of decrementing position by a size of some object.

6.29.3.9 Position Position::operator- (const Vector & *vec*)

Parameters:

vec vector to substract

Result of modificating position by a vector.

6.29.3.10 Position & Position::operator-= (const Position & *pos*)

Parameters:

pos substraction target

Simple assignment by substraction.

6.29.3.11 Position & Position::operator-= (const Size & *size*)

Parameters:

size size to decrement by

Decrementation of position by a size of some object.

6.29.3.12 Position & Position::operator-= (const Vector & *vec*)

Parameters:

vec vector to substract

Modification of position by a vector.

6.29.4 Member Data Documentation

6.29.4.1 Uint Scr::Position::row

row number

6.29.4.2 Uint Scr::Position::col

column number

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

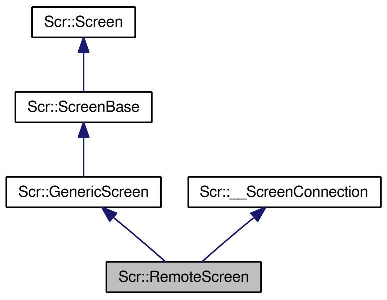
- include/rexio/commons.h++
- lib/screen/src/core/commons.c++

6.30 Scr::RemoteScreen Class Reference

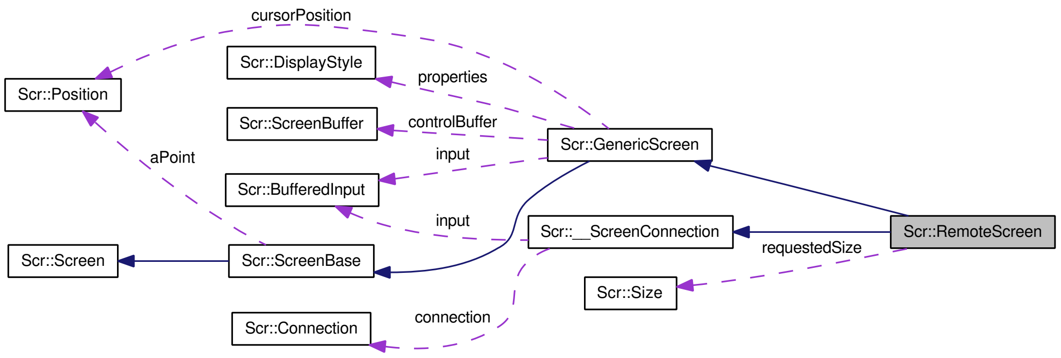
TELNET connection.

`#include <remotescreen.h++>`

Inheritance diagram for Scr::RemoteScreen:



Collaboration diagram for Scr::RemoteScreen:



Public Member Functions

- virtual const char ∗ GetType () const throw (TerminalTypeUnknown)
- int ProcessConnection ()

Protected Member Functions

- void AnswerCommand ()
- void SubnegotiateWindowSize ()

- void SubnegotiateTerminalType ()
- virtual Key DecodeKeyPressedHandleTelnet ()

Protected Attributes

- Size requestedSize
- bool resizeRequestPending
- char counter

6.30.1 Detailed Description

TELNET connection.

6.30.2 Member Function Documentation

6.30.2.1 void Scr::RemoteScreen::AnswerCommand () `[protected]`

general subnegotiation switch.

6.30.2.2 void Scr::RemoteScreen::SubnegotiateWindowSize () `[protected]`

Read window size and possibly call OnResize; Handle subnegotiation end (SE) correctly. ASSUME, that IAC SB NAWS was just recv, so process w h IAC SE (check for correctnes after each).

6.30.2.3 void Scr::RemoteScreen::SubnegotiateTerminalType () `[protected]`

read information on terminal type.

6.30.2.4 Scr::Key Scr::RemoteScreen::DecodeKeyPressedHandleTelnet () `[protected, virtual]`

Process characters according to telnet protocol. Handle variants of Enter key.

6.30.2.5 const char * Scr::RemoteScreen::GetType () const throw (TerminalTypeUnknown) `[virtual]`

Returns:

returns information retrieved by SubnegotiateTerminalType() if telnet client supports TELNET::TTYPE extension (RFC 1091). If client does not support this feature, dumb terminal type will be assumed and NULL will be returned. "unknown" special value will be returned

Reimplemented from Scr::GenericScreen.

6.30.2.6 int Scr::RemoteScreen::ProcessConnection () `[virtual]`

Returns:

value of exitcode, as it was in the moment of connection termination if successful.

Initialize, conduct and end connection in way apropriate to connection type, operating system etc. Inform Scr::Connection object supplied about all captured events

Note:

> as function (for design reasons) lacks exception-set specification, it may throw any exceptions, but it is recommended, that only exceptions typical to Scr::Connection::Start() will be thrown.

Implements Scr::__ScreenConnection.

6.30.3 Member Data Documentation

6.30.3.1 Size Scr::RemoteScreen::requestedSize `[protected]`

When resize request is pending, store requested dimensions here.

6.30.3.2 bool Scr::RemoteScreen::resizeRequestPending `[protected]`

Client has requested resize. Let him wait until counter == 0.

6.30.3.3 char Scr::RemoteScreen::counter `[protected]`

1-2-3-...-254-255-0

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

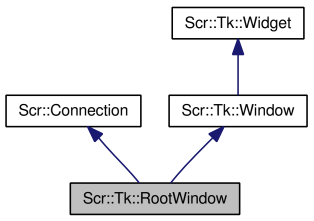
- lib/screen/include/remotescreen.h++
- lib/screen/src/real/remotescreen.c++

6.31 Scr::Tk::RootWindow Class Reference

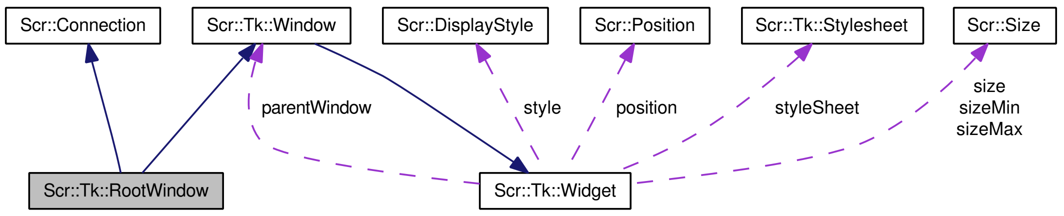
Main application window.

`#include <rootwindow.h++>`

Inheritance diagram for Scr::Tk::RootWindow:



Collaboration diagram for Scr::Tk::RootWindow:



Public Member Functions

- virtual int Start (int argc, char ∗∗argv) throw (StartFailed,Screen::IllegalCharacter)
- virtual int Start () throw (StartFailed,Screen::IllegalCharacter)
- virtual RootWindow & GetRootWindow () throw ()
- virtual void OnRedraw (Screen &screen) throw ()
- void LoadStylesheet (const char ∗filename) throw (FileNotOpened, Stylesheet::ParsingError)
- void ForceRepaint () throw ()
- void ForceOnRedraw () throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Member Functions

- RootWindow (std::istream &_input, std::ostream &_output, const DisplayStyle &_-style=ROOTWINDOW_DEFAULT_STYLE) throw ()
- virtual Screen & GetScreen () throw ()
- virtual Uint GetAbsoluteColumn () throw ()
- virtual Uint GetAbsoluteRow () throw ()

6.31.1 Detailed Description

Main application window.

Lord of the widgets and the main connection point between the Toolkit and lower level library.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 RootWindow::RootWindow (std::istream & *_input*, std::ostream & *_output*, const DisplayStyle & *_style* = `ROOTWINDOW_DEFAULT_STYLE`) throw () `[protected]`

Parameters:

_input input stream handler

_output output stream handler

_style default style

The input handlers make it possible to attach to any character device. Specifically it can be an ordinary terminal or a tcp connection to a remote telnet application.

6.31.3 Member Function Documentation

6.31.3.1 Scr::Screen & RootWindow::GetScreen () throw () `[protected, virtual]`

Returns:

Screen handler reference.

Reimplemented from Scr::Tk::Window.

6.31.3.2 Scr::Uint RootWindow::GetAbsoluteColumn () throw () `[protected, virtual]`

Returns:

0

Reimplemented from Scr::Tk::Window.

6.31.3.3 Scr::Uint RootWindow::GetAbsoluteRow () throw () `[protected, virtual]`

Returns:

0

Reimplemented from Scr::Tk::Window.

6.31.3.4 int RootWindow::Start (int *argc*, char ** *argv*) throw (StartFailed,Screen::IllegalCharacter) `[virtual]`

Parameters:

argc number of arguments
argv C-style array of arguments

Start connection. *argv* can be parsed in inheritting classes.

See also:

Start() for detailed info

RootWindow specific:

Arguments: -style=FILE - Use this FILE as a stylesheet.

Reimplemented from Scr::Connection.

6.31.3.5 int RootWindow::Start () throw (StartFailed,Screen::IllegalCharacter) `[virtual]`

Returns:

result of whole connection. If broken, the result is 1. Else the result is argument passed to Exit(int)

Start connection (with no arguments - they must be set with application specific methods defined by programmer). Function blocks execution of thread up to finish of connection.

Exceptions:

Scr::Connection::AlreadyRunning when connection has already been started (one execution thread per class instance allowed) and hasn't yet been stopped.

Scr::Connection::Broken is thrown when connection is broken (i.e. input/output error occured)

Scr::Connection::FailedToStart when connecction can not be estabilished for some reason.

Note:

as Start() is defined in way, that allows it to throw only one exception class and all OnEvent functions do not allow any exceptions, all of them must be handled within exception handling function. Unexpected exception handler will be used otherwise.

Reimplemented from Scr::Connection.

6.31.3.6 RootWindow & RootWindow::GetRootWindow () throw () `[virtual]`

Returns:

referene to self

Reimplemented from Scr::Tk::Window.

6.31.3.7 void RootWindow::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Window.

6.31.3.8 void RootWindow::LoadStylesheet (const char ∗ *filename*) throw (FileNotOpened, Stylesheet::ParsingError)

Parameters:

filename location of the stylesheet

Loads stylesheet from the given location.

Exceptions:

FileNotOpened is thrown if the file couldn't be opened.

ParsingError is thrown if the input file contained inappropriate input.

6.31.3.9 void RootWindow::ForceRepaint () throw ()

Repaints whole screen (useful after invoking background programs, that modify its content)

6.31.3.10 void Scr::Tk::RootWindow::ForceOnRedraw () throw () `[inline]`

Trigger OnRedraw event

6.31.3.11 virtual bool Scr::Tk::RootWindow::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::Window.

6.31.3.12 virtual const char∗ Scr::Tk::RootWindow::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::Window.

6.31.3.13 virtual const char∗ Scr::Tk::RootWindow::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::Window.

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

- include/rexio/tk/rootwindow.h++
- lib/toolkit/src/rootwindow.c++

6.32 Scr::RScreen< LOCATION, TYPE > Class Template Reference

template class representing full implementation of Scr::Screen and Scr::__ScreenConnection

`#include <core.h++>`

6.32.1 Detailed Description

template<class LOCATION, class TYPE> class Scr::RScreen< LOCATION, TYPE >

template class representing full implementation of Scr::Screen and Scr::__ScreenConnection

Parameters:

LOCATION local, telnet etc ..

TYPE frameless VT100-like, UTF8, Windows....

See figure attached to Scr:: namespace description for more details

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

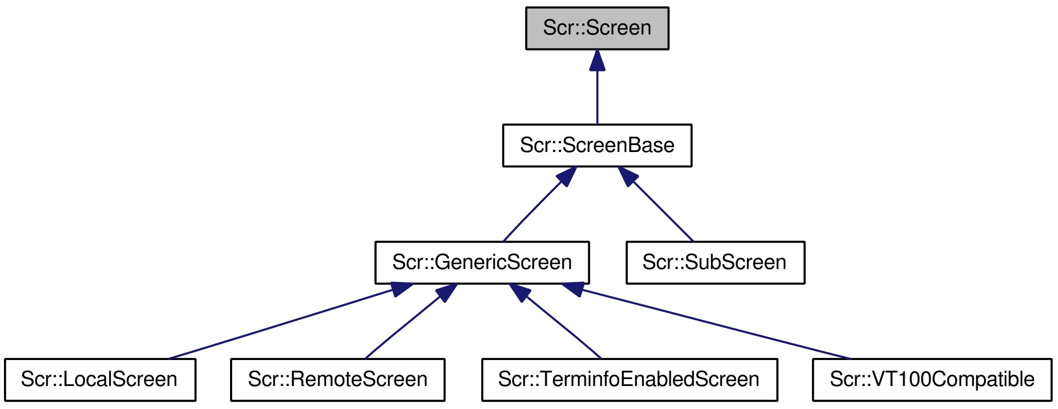
- lib/screen/include/core.h++

6.33 Scr::Screen Class Reference

Class representing basic output operation is defined as ABC (abstract base).

`#include <screen.h++>`

Inheritance diagram for Scr::Screen:



Public Member Functions

- virtual void Clear ()=0 throw ()
- virtual void SetBgColor (Bg::Color col)=0 throw ()
- virtual void SetFgColor (Fg::Color col)=0 throw ()
- virtual void SetFgStyle (Fg::Style s)=0 throw ()
- virtual void GotoYX (Uint y, Uint x)=0 throw (GotoOutOfRange)
- virtual void AddCharacter (char c)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddCharacter (wchar_t c)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const char ∗text)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const std::string &text)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const wchar_t ∗text)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const std::wstring &text)=0 throw (PrintOutOfRange, IllegalCharacter)

- virtual Uint AddTextCols (const wchar_t ∗text, Uint limitcols)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual Uint AddTextCols (const std::wstring &text, Uint limitcols)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void HorizontalLine (char c, Uint n)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void HorizontalLine (wchar_t c, Uint n)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void VerticalLine (char c, Uint n)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void VerticalLine (wchar_t c, Uint n)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void Rectangle (char c, const Size &s)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void Rectangle (wchar_t c, const Size &s)=0 throw (PrintOutOfRange, IllegalCharacter)
- virtual void Resize (Uint rows, Uint cols)=0 throw (IllegalOperation)
- virtual void ForceCursorPosition (Position p)=0 throw (RangeError)
- virtual void HideCursor ()=0 throw (CursorVisibilityNotSupported)
- virtual void ShowCursor ()=0 throw (CursorVisibilityNotSupported)
- virtual void Refresh ()=0 throw (ConnectionError)
- virtual Screen ∗ CreateSubScreen (Uint _y_offset, Uint _x_offset, Uint _h, Uint _w)=0 throw (SubscreenOutOfRange)
- virtual const char ∗ GetType () const =0 throw (TerminalTypeUnknown)
- virtual Uint GetY () const =0 throw ()
- virtual Uint GetX () const =0 throw ()
- virtual Uint GetHeight () const =0 throw ()
- virtual Uint GetWidth () const =0 throw ()
- virtual bool GetCursorVisibility () const =0 throw ()

6.33.1 Detailed Description

Class representing basic output operation is defined as ABC (abstract base).

Operations are performed using subroutines apropriate to output type. Note, that some implementations of Screen (i.e. remote ones) use spcific forms of compression to limit data transfer, other rather optimize CPU usage.

6.33.2 Member Function Documentation

6.33.2.1 virtual void Scr::Screen::Clear () throw () `[pure virtual]`

Fill whole screen with current background colour.

Note:

function does not operate on physical screen. Use Refresh to see effect.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.2 virtual void Scr::Screen::SetBgColor (Bg::Color *col*) throw () `[pure virtual]`

Parameters:

col new background colour to be set

Returns:

nothing upon successful execution

Function sets background colour. Background colour is of type Bg::Color. Typical use example: `myscreen.SetBgColor(Bg::Black) .`

Function is exception safe as it does not throw any exceptions.

Note:

thanks to overloaded operator <<, something like `myscreen << Bg::Black` will also be valid and will do exactly the same as above.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.3 virtual void Scr::Screen::SetFgColor (Fg::Color *col*) throw () `[pure virtual]`

Parameters:

col new foreground colour to be set

Returns:

nothing upon successful execution

Function sets foreground colour. Background colour is of type Bg::Color. Typical use example: `myscreen.SetFgColor(Fg::Red).`

Function is exception safe as it does not throw any exceptions.

Note:

thanks to overloaded operator <<, something like `myscreen << Fg::Red` will also be valid and will do exactly the same as above.
colour is not only foreground property: Fg style sets bright or dark variant of each colour, and it doubles total number of availble colours.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.4 virtual void Scr::Screen::SetFgStyle (Fg::Style *s*) throw () `[pure virtual]`

Parameters:

s new foreground text style to be set

Returns:

nothing upon successful execution

Set foreground style (i.e. bright (bold) or dim (regular)). Maybe once upon the time more styles will be suppotred to utilise capabilities of more advanced terminal types (such as blink and underline for DEC VT220), but for now we don't specify this, as portability is one of primary goals for our library

Function is exception safe as it does not throw any exceptions.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.5 virtual void Scr::Screen::GotoYX (Uint *y*, Uint *x*) throw (GotoOutOfRange) `[pure virtual]`

Parameters:

y

x new coordinates of active point (please remember the order of theese attributes)

Change active point position (that is point, where writing will start after invocation of AddText or AddCharacter.

Function throws exception Scr::Screen::GotoOutOfRange when coordinates exceed size of screen. After exception throw active position is undefined.

See also:

SetFgColor # SetBgColor

Returns:

nothing upon successful execution

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.6 virtual void Scr::Screen::AddCharacter (char *c*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

c character to be printed

Returns:

nothing upon successful execution

Print single low ascii character (for characters out of basic 7-bit ascii set please use integer version of this function and proper UNICODE codes of characters)

Exceptions:

Scr::Screen::PrintOutOfRange as for AddText

Scr::Screen::IllegalCharacter negative signed (or over-127-unsigned) c supplied.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.7 virtual void Scr::Screen::AddCharacter (wchar_t *c*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

c character to be printed

Returns:

nothing upon successful execution

Print single unicode character.

Note:

what programmes supply as parameter is direct number of character, not UTF-8 encoded version of it. UTF-8 may be supplied using AddText

Exceptions:

Scr::Screen::PrintOutOfRange as for AddText

Scr::Screen::IllegalCharacter too large value of c.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.8 virtual void Scr::Screen::AddText (const char ∗ *text*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

text traditional null-terminated string in UTF-8 encoding.

Returns:

nothing upon successful execution

Adds specified text in position starting from active point (see GotoYX). Moves active point just after the newly added text irrespectively if this position is valid (so next text will start just after it, always in the same line). Function does not support line breaks.

As function supports UTF-8, it also requires string to be valid UTF-8, so each character must be low ascii (1-127) or multibyte.

Note:

function will not emit text to physical screen, unless Refresh called afterwards

Exceptions:

Scr::Screen::PrintOutOfRange is thrown if initial position of active point is invalid, or if text is too long (as function does not support line breaks).

If the text ends exactly in last column of screen, active point is set after it, in the same line, so is invalid, and next trial of usage of this function (or any other character-adding one) will fail with Scr::Screen::PrintOutOfRange.

Scr::Screen::IllegalCharacter will be thrown if text supplied is not a valid UTF-8 string (even "overlong sequences" will be considered illegal (according to an apropriate RFC

See also:

AddCharacter, Refresh, RFC 3629

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.9 virtual void Scr::Screen::AddText (const std::string & *text*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

text as above but as std::string, not C-style string

exceptions: as above.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.10 virtual void Scr::Screen::AddText (const wchar_t ∗ *text*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

text wide UNICODE string to be printed

Exceptions:

PrintOutOfRange is thrown if initial position of active point is invalid, or if text is too long (as function does not support line breaks).

IllegalCharacter will be thrown if text supplied is not a valid UTF-8 string (even "overlong sequences" will be considered illegal (according to an apropriate RFC

Parameters:

text text to be printed

See also:

Screen::AddText(const char ∗ text) for extensive description

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.11 virtual void Scr::Screen::AddText (const std::wstring & *text*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

text text to be printed

See also:

Screen::AddText(const char ∗ text) for extensive description

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.12 virtual Uint Scr::Screen::AddTextCols (const wchar_t ∗ *text*, Uint *limitcols*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

text wide string

limitcols max width in columns

Function prints AT MOST limitcols wide string. Width means number of columns, which is not the same thing as number of characters, as most CJK glyphs are multicolumn.

Exceptions:

PrintOutOfRange is thrown if initial position of active point is invalid, or if text is too long (as function does not support line breaks).

IllegalCharacter will be thrown if text supplied is not a valid UTF-8 string (even "overlong sequences" will be considered illegal (according to an apropriate RFC

See also:

Screen::AddText(const char ∗ text) for extensive description

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.13 virtual Uint Scr::Screen::AddTextCols (const std::wstring & *text*, Uint *limitcols*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

text wide string

limitcols max width in columns

Function prints AT MOST limitcols wide string. Width means number of columns, which is not the same thing as number of characters, as most CJK glyphs are multicolumn.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.14 virtual void Scr::Screen::HorizontalLine (char *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

c ASCII character

n number of repetitions (length of line)

Function adds horizontal line of n characters c.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.15 virtual void Scr::Screen::HorizontalLine (wchar_t *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

c UNICODE character

n number of repetitions (length of line)

Function adds horizontal line of n characters c.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.16 virtual void Scr::Screen::VerticalLine (char *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

- ***c*** ASCII character
- ***n*** number of repetitions (length of line)

Function adds verticel line of n characters c.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.17 virtual void Scr::Screen::VerticalLine (wchar_t *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

- ***c*** UNICODE character
- ***n*** number of repetitions (length of line)

Function adds vertical line of n characters c.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.18 virtual void Scr::Screen::Rectangle (char *c*, const Size & *s*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

- ***c*** character used to create rectangle
- ***s*** dimensions of rectangle

Function creates rectangle of characters. s specifies number of rows and number of repetitions of character c in each row.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.19 virtual void Scr::Screen::Rectangle (wchar_t *c*, const Size & *s*) throw (PrintOutOfRange, IllegalCharacter) `[pure virtual]`

Parameters:

- ***c*** character used to create rectangle
- ***s*** dimensions of rectangle

Function creates rectangle of characters. s specifies number of rows and number of repetitions of character c in each row.

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.20 virtual void Scr::Screen::Resize (Uint *rows*, Uint *cols*) throw (IllegalOperation) `[pure virtual]`

Parameters:

rows new number of rows (new height) of screen

cols new number of columns of screen

Returns:

nothing upon successful execution

Change the output size.

Note:

this function does not change size of physical screen, it may only be used to resize this object to fit physical screen size. If screen grows, new characters are filled with current background colour.
Function does not refresh the physical screen after it's resizing, so it's content is undefined after call of this function (however left-top part of it will be restored after Refresh call).

Exceptions:

Scr::Screen::Exception::IllegalOperation if particular screen may nor be resized for some implementation- or platform- specific reasons. In particular case primitive subscreens may not be resized (SubscreenResize specialization of exception is thrown then).

Implemented in Scr::GenericScreen, Scr::SubScreen, Scr::TerminfoEnabledScreen, and Scr::VT100Compatible.

6.33.2.21 virtual void Scr::Screen::ForceCursorPosition (Position *p*) throw (RangeError) `[pure virtual]`

Parameters:

p new cursor position

Force cursor position after finishing next refresh. If ∗this is a subscreen, position (relative to ∗this) will be mapped to the physical screen.

Note:

effective position after refresh will be position set by last successful call to ForceCursorPosition

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.22 virtual void Scr::Screen::HideCursor () throw (CursorVisibilityNotSupported) `[pure virtual]`

make cursor invisible

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.23 virtual void Scr::Screen::ShowCursor () throw (CursorVisibilityNotSupported) `[pure virtual]`

make it visible again

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.24 virtual void Scr::Screen::Refresh () throw (ConnectionError) `[pure virtual]`

Returns:

nothing upon successful execution

Rewrite internal buffers to physical screen. When writing complex, multi-layer items, it is recommended to call this function after finishing writing everything. When small changes need to be displayed, it may be called every single AddCharacter, as it can't be a very expansive operation in terms of CPU or transfer usage (remote implementations will be optimized for transfer, while local will be writen to achieve best performance for specific terminal).

Implemented in Scr::GenericScreen, Scr::SubScreen, Scr::TerminfoEnabledScreen, and Scr::VT100Compatible.

6.33.2.25 virtual Screen∗ Scr::Screen::CreateSubScreen (Uint *_y_offset*, Uint *_x_offset*, Uint *_h*, Uint *_w*) throw (SubscreenOutOfRange) `[pure virtual]`

Parameters:

_y_offset vertical offset from top edge of this screen to top edge of new SubScreen.

_x_offset horizontal offser

_h height

_w with

Returns:

pointer to new SubScreen (programmer will have to free it's resources to prevent memory leak and other errors).

Exceptions:

Scr::Screen::SubscreenOutOfRange is thrown when too big subscreen requested or inapropriate position specified

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.26 virtual const char∗ Scr::Screen::GetType () const throw (TerminalTypeUnknown) `[pure virtual]`

Returns:

current type of terminal

Implemented in Scr::GenericScreen, Scr::LocalScreen, Scr::RemoteScreen, and Scr::SubScreen.

6.33.2.27 virtual Uint Scr::Screen::GetY () const throw () `[pure virtual]`

Returns:

vertical offset of active point

Implemented in Scr::ScreenBase.

6.33.2.28 virtual Uint Scr::Screen::GetX () const throw () `[pure virtual]`

Returns:

horizontal offset of active point

Implemented in Scr::ScreenBase.

6.33.2.29 virtual Uint Scr::Screen::GetHeight () const throw () `[pure virtual]`

Returns:

Current height of screen

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.30 virtual Uint Scr::Screen::GetWidth () const throw () `[pure virtual]`

Returns:

Current width of screen

Implemented in Scr::GenericScreen, and Scr::SubScreen.

6.33.2.31 virtual bool Scr::Screen::GetCursorVisibility () const throw () `[pure virtual]`

Returns:

true if cursor is visible, false if it ishidden

See also:

ShowCursor HideCursor

Implemented in Scr::GenericScreen, and Scr::SubScreen.

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

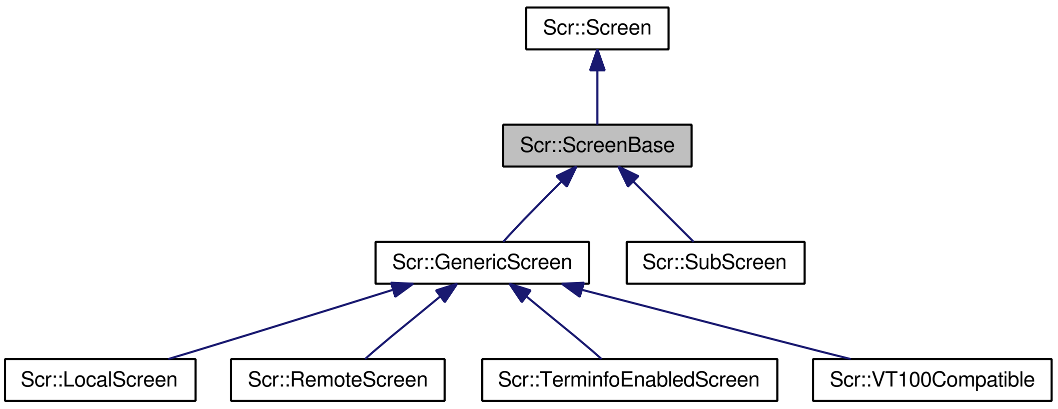
- include/rexio/screen.h++
- lib/screen/src/core/screen.c++

6.34 Scr::ScreenBase Class Reference

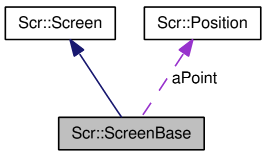
Implements features common to subscreen and generic screen.

`#include <screenbase.h++>`

Inheritance diagram for Scr::ScreenBase:



Collaboration diagram for Scr::ScreenBase:



Public Member Functions

- Uint GetX () const throw ()
- Uint GetY () const throw ()

Protected Attributes

- Position aPoint

6.34.1 Detailed Description

Implements features common to subscreen and generic screen.

6.34.2 Member Function Documentation

6.34.2.1 Uint Scr::ScreenBase::GetX () const throw () `[virtual]`

Returns:

horizontal offset from the left edge of the screen

Implements Scr::Screen.

6.34.2.2 Uint Scr::ScreenBase::GetY () const throw () `[virtual]`

Returns:

vertical offset from top of the screen

Implements Scr::Screen.

6.34.3 Member Data Documentation

6.34.3.1 Position Scr::ScreenBase::aPoint `[protected]`

vertical and horizontal offset from the left edge of the screen

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

- lib/screen/include/screenbase.h++
- lib/screen/src/core/screenbase.c++

6.35 Scr::ScreenBuffer Class Reference

buffer of characters, supporting colours and unicode.

`#include <screenbuffer.h++>`

Public Member Functions

- ScreenBuffer (Uint _rows, Uint columns, const ScreenCharacter &character=ScreenCharacter(' ', DisplayStyle(Fg::White, Fg::Dark, Bg::Black)))
- ScreenRow & operator[ ] (Uint _i)
- ScreenBuffer & operator= (const ScreenBuffer &other)
- bool operator== (const ScreenBuffer &other)
- bool operator!= (const ScreenBuffer &other)
- void Resize (Uint newHeight, Uint newWidth, const ScreenCharacter &character=ScreenCharacter(' ', DisplayStyle(Fg::White, Fg::Dark, Bg::Black)))
- Uint GetHeight () const
- Uint GetWidth () const
- void Fill (const ScreenCharacter &character)

6.35.1 Detailed Description

buffer of characters, supporting colours and unicode.

Class represents character buffer.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 ScreenBuffer::ScreenBuffer (Uint *_rows*, Uint *columns*, const ScreenCharacter & *character* = ScreenCharacter(' ', DisplayStyle(Fg::White, Fg::Dark,Bg::Black)))

Parameters:

_rows initial height of screen buffer

columns initial width of screen buffer

character initial fill of screen buffer (by default plain black background (filled with space))

Note:

buffer size may be changed runtime.

6.35.3 Member Function Documentation

6.35.3.1] ScreenRow& Scr::ScreenBuffer::operator[ ] (Uint *_i*) [inline]

Parameters:

_i row number (0..height-1)

Returns:

reference to specific row

Note:

no range checking, and no exception-connected warranties for this function.

6.35.3.2 ScreenBuffer & ScreenBuffer::operator= (const ScreenBuffer & *other*)

Parameters:

other right-hand operand

Assign other screen to this one. Function copies whole contents, so complexity is O(width∗height).

6.35.3.3 bool ScreenBuffer::operator== (const ScreenBuffer & *other*)

Parameters:

other right-hand operand

Returns:

true if size of each buffer is equal, each character equals its counterpart on second buffer, both in terms of unicode value and colour.

6.35.3.4 bool ScreenBuffer::operator!= (const ScreenBuffer & *other*)

Parameters:

other right-hand operand

Returns:

true if any difference occours between two screens.

6.35.3.5 void ScreenBuffer::Resize (Uint *newHeight*, Uint *newWidth*, const ScreenCharacter & *character* = ScreenCharacter(' ', DisplayStyle(Fg::White, Fg::Dark,Bg::Black)))

Parameters:

newHeight new height of screen buffer

newWidth new width of screen buffer

character character, to fill new rows or colums (if their number grows) with.

6.35.3.6 Uint ScreenBuffer::GetHeight () const

Returns:

current height of buffer (number of rows)

6.35.3.7 Uint ScreenBuffer::GetWidth () const

Returns:

current width of buffer (number of characters in each row)

6.35.3.8 void ScreenBuffer::Fill (const ScreenCharacter & *character*)

Parameters:

character character

Function fills whole buffer with specific character.

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

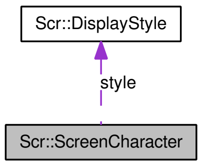
- lib/screen/include/screenbuffer.h++
- lib/screen/src/core/screenbuffer.c++

6.36 Scr::ScreenCharacter Class Reference

character to be displayed with all it's properties

`#include <screenbuffer.h++>`

Collaboration diagram for Scr::ScreenCharacter:



Public Member Functions

- ScreenCharacter (Uint _c, const DisplayStyle &_style)
- ScreenCharacter & operator= (const ScreenCharacter &other)
- bool operator== (const ScreenCharacter &other)
- bool operator!= (const ScreenCharacter &other)

6.36.1 Detailed Description

character to be displayed with all it's properties

6.36.2 Constructor & Destructor Documentation

6.36.2.1 ScreenCharacter::ScreenCharacter (Uint *_c*, const DisplayStyle & *_style*)

Parameters:

_c character UNICODE code

_style colour etc.

6.36.3 Member Function Documentation

6.36.3.1 ScreenCharacter & ScreenCharacter::operator= (const ScreenCharacter & *other*)

Parameters:

other right-hand operand

Assignment operator copies character and all it's properties

6.36.3.2 bool ScreenCharacter::operator== (const ScreenCharacter & *other*)

Parameters:

other right-hand operand

Comparison operator returns true if colour and character match

6.36.3.3 bool ScreenCharacter::operator!= (const ScreenCharacter & *other*)

Parameters:

other right-hand operand

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

- lib/screen/include/screenbuffer.h++
- lib/screen/src/core/screenbuffer.c++

6.37 Scr::ScreenRow Class Reference

single row of ScreenBuffer object (which may contain more rows)

`#include <screenbuffer.h++>`

Public Member Functions

- ScreenRow & operator= (const ScreenRow &other)
- ScreenCharacter & operator[ ] (Uint i)
- bool operator== (const ScreenRow &other)
- bool operator!= (const ScreenRow &other)

Protected Member Functions

- ScreenRow (Uint width, const ScreenCharacter &character=ScreenCharacter(' ', DisplayStyle(Fg::White, Fg::Dark, Bg::Black)))
- void Resize (Uint newWidth, const ScreenCharacter &character)

6.37.1 Detailed Description

single row of ScreenBuffer object (which may contain more rows)

Class implements single row of characters, so it encapsulates std::vector.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 ScreenRow::ScreenRow (Uint *width*, const ScreenCharacter & *character* = ScreenCharacter(' ', DisplayStyle(Fg::White, Fg::Dark,Bg::Black))) `[protected]`

Parameters:

width number of characters

character initial content

6.37.3 Member Function Documentation

6.37.3.1 void ScreenRow::Resize (Uint *newWidth*, const ScreenCharacter & *character*) `[protected]`

Parameters:

newWidth new width of specific row.

character if new width is greater, than current, additional fields will be filled with this specific character

Note:

declared as protected function to prevent changing width outside of ScreenBuffer, and therefore to assure, that buffer will be rectangular (equal width for each row).

6.37.3.2 ScreenRow & ScreenRow::operator= (const ScreenRow & *other*)

Parameters:

other right-hand operand

copy content of one buffer to second one. If size differs, target is resized to match source.

6.37.3.3] ScreenCharacter& Scr::ScreenRow::operator[ ] (Uint *i*) `[inline]`

Parameters:

i index

Array element access operator returns reference to the specific character.

6.37.3.4 bool ScreenRow::operator== (const ScreenRow & *other*)

Parameters:

other right-hand operand

Comparison for equal compares each character, and returns true if no difference found

6.37.3.5 bool ScreenRow::operator!= (const ScreenRow & *other*)

Parameters:

other right-hand operand

Comparison for equal compares each character, and returns true if any difference found

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

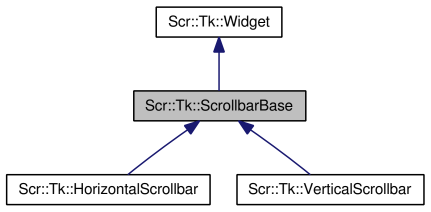
- lib/screen/include/screenbuffer.h++
- lib/screen/src/core/screenbuffer.c++

6.38 Scr::Tk::ScrollbarBase Class Reference

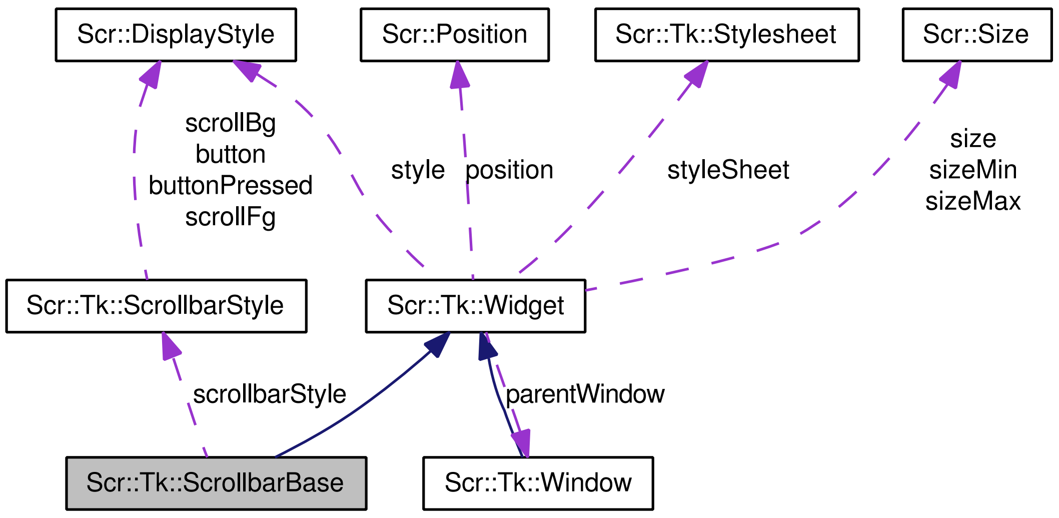
Base for implementing scrollbars.

`#include <scrollbar.h++>`

Inheritance diagram for Scr::Tk::ScrollbarBase:



Collaboration diagram for Scr::Tk::ScrollbarBase:



Public Member Functions

- virtual void OnRedraw (Screen &screen)=0 throw ()
- virtual void SetScrollSize (Uint _scrollSize) throw ()
- virtual Uint GetScrollSize () const throw ()
- virtual void SetScrollOffset (Uint _scrollOffset) throw ()
- virtual Uint GetScrollOffset () const throw ()
- virtual void SetScrollProgress (float progress) throw ()
- virtual float GetScrollProgress () const throw ()
- virtual void SetScrollbarStyle (const ScrollbarStyle &_scrollStyle) throw ()
- virtual const ScrollbarStyle & GetScrollbarStyle () const throw ()
- virtual void SetStylesheet (Stylesheet ∗_styleSheet) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

6.38.1 Detailed Description

Base for implementing scrollbars.

This class implements interface for HorizontalScrollbar and VerticalScrollbar. Allows setting progress, offsets, size, style.

6.38.2 Member Function Documentation

6.38.2.1 virtual void Scr::Tk::ScrollbarBase::OnRedraw (Screen & *screen*) throw () `[pure virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Widget.

Implemented in Scr::Tk::HorizontalScrollbar, and Scr::Tk::VerticalScrollbar.

6.38.2.2 void ScrollbarBase::SetScrollSize (Uint *_scrollSize*) throw () `[virtual]`

Parameters:

_scrollSize Set virtual area that the scrollbar should cover.

6.38.2.3 Uint ScrollbarBase::GetScrollSize () const throw () `[virtual]`

Returns:

virtual size the scrollbar covers.

6.38.2.4 void ScrollbarBase::SetScrollOffset (Uint *_scrollOffset*) throw () `[virtual]`

Parameters:

_scrollOffset Set number of virtual offset.

6.38.2.5 Uint ScrollbarBase::GetScrollOffset () const throw () `[virtual]`

Return virtual offset.

6.38.2.6 void ScrollbarBase::SetScrollProgress (float *progress*) throw () `[virtual]`

Parameters:

progress Provided for convenience. Sets the scrollOffset in respect to scrollSize accordingly to given progress.

6.38.2.7 float ScrollbarBase::GetScrollProgress () const throw () `[virtual]`

Returns:

Current scrolling progress.

6.38.2.8 void ScrollbarBase::SetScrollbarStyle (const ScrollbarStyle & *_scrollStyle*) throw () `[virtual]`

Parameters:

_scrollStyle new style Set scrollbar specific style.

6.38.2.9 const ScrollbarStyle & ScrollbarBase::GetScrollbarStyle () const throw () `[virtual]`

Returns:

current scrollbar specific style

6.38.2.10 virtual void Scr::Tk::ScrollbarBase::SetStylesheet (Stylesheet ∗ *_styleSheet*) throw () `[inline, virtual]`

Parameters:

_styleSheet pointer to style data

Apply Stylesheet to this widget. Reinitialize any style properties if their alternatives are supplied.

Reimplemented from Scr::Tk::Widget.

6.38.2.11 virtual bool Scr::Tk::ScrollbarBase::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::HorizontalScrollbar, and Scr::Tk::VerticalScrollbar.

6.38.2.12 virtual const char∗ Scr::Tk::ScrollbarBase::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::HorizontalScrollbar, and Scr::Tk::VerticalScrollbar.

6.38.2.13 virtual const char∗ Scr::Tk::ScrollbarBase::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::HorizontalScrollbar, and Scr::Tk::VerticalScrollbar.

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

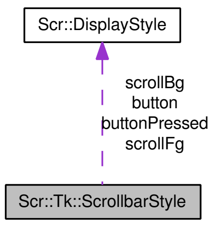
- include/rexio/tk/scrollbar.h++
- lib/toolkit/src/scrollbar.c++

6.39 Scr::Tk::ScrollbarStyle Struct Reference

Scrollbars specific style.

`#include <scrollbar.h++>`

Collaboration diagram for Scr::Tk::ScrollbarStyle:



Public Member Functions

- ScrollbarStyle (const DisplayStyle &_button=_DEFAULT_SCROLLBAR_BUTTON, const DisplayStyle &_buttonPressed=_DEFAULT_SCROLLBAR_BUTTONPRESSED, wchar_t _buttonUp=_DEFAULT_SCROLLBAR_BUTTONUP, wchar_t _buttonDown=_DEFAULT_-SCROLLBAR_BUTTONDOWN, wchar_t _buttonLeft=_DEFAULT_SCROLLBAR_-BUTTONLEFT, wchar_t _buttonRight=_DEFAULT_SCROLLBAR_BUTTONRIGHT, const DisplayStyle &_scrollBg=_DEFAULT_SCROLLBAR_SCROLLBG, wchar_t _-scrollField=_DEFAULT_SCROLLBAR_SCROLLFIELD, const DisplayStyle &_scrollFg=_-DEFAULT_SCROLLBAR_SCROLLFG, wchar_t _scrollHandleV=_DEFAULT_SCROLLBAR_-SCROLLHANDLEV, wchar_t _scrollHandleH=_DEFAULT_SCROLLBAR_SCROLLHANDLEH) throw ()

Public Attributes

- DisplayStyle button

  style for directional buttons

- DisplayStyle buttonPressed

  style for pressed buttons

- wchar_t buttonUp

  symbol for drawing up button

- wchar_t buttonDown

  symbol for drawing down button

- wchar_t buttonLeft

  symbol for drawing left button

- wchar_t buttonRight

  symbol for drawing right button

- DisplayStyle scrollBg

  style for drawing scrollbar's

- wchar_t scrollField

  symbol for drawing scrollbar's area

- DisplayStyle scrollFg

  style for drawing scrollbar's area

- wchar_t scrollHandleV

  symbol for vertical handle

- wchar_t scrollHandleH

  symbol for horizontal handle

6.39.1 Detailed Description

Scrollbars specific style.

Describes the way a specific scrollbar is drawn.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 Scr::Tk::ScrollbarStyle::ScrollbarStyle (const DisplayStyle & *_button* = `_DEFAULT_-SCROLLBAR_BUTTON`, const DisplayStyle & *_buttonPressed* = `_DEFAULT_SCROLLBAR_-BUTTONPRESSED`, wchar_t *_buttonUp* = `_DEFAULT_SCROLLBAR_BUTTONUP`, wchar_t *_-buttonDown* = `_DEFAULT_SCROLLBAR_BUTTONDOWN`, wchar_t *_buttonLeft* = `_DEFAULT_-SCROLLBAR_BUTTONLEFT`, wchar_t *_buttonRight* = `_DEFAULT_SCROLLBAR_BUTTONRIGHT`, const DisplayStyle & *_scrollBg* = `_DEFAULT_SCROLLBAR_SCROLLBG`, wchar_t *_scrollField*

= `_DEFAULT_SCROLLBAR_SCROLLFIELD`, **const DisplayStyle &** ***_scrollFg*** **=** `_DEFAULT_-SCROLLBAR_SCROLLFG`, **wchar_t** ***_scrollHandleV*** **=** `_DEFAULT_SCROLLBAR_SCROLLHANDLEV`, **wchar_t** ***_scrollHandleH*** **=** `_DEFAULT_SCROLLBAR_SCROLLHANDLEH`**) throw ()** `[inline]`

Parameters:

_button style for directional buttons

_buttonPressed style for pressed buttons

_buttonUp symbol for drawing up button

_buttonDown symbol for drawing down button

_buttonLeft symbol for drawing left button

_buttonRight symbol for drawing right button

_scrollBg style for drawing scrollbar's

_scrollField symbol for drawing scrollbar's area

_scrollFg style for drawing scrollbar's area

_scrollHandleV symbol for vertical handle

_scrollHandleH symbol for horizontal handle

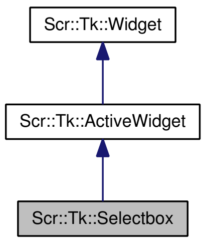
The documentation for this struct was generated from the following file:

- include/rexio/tk/scrollbar.h++

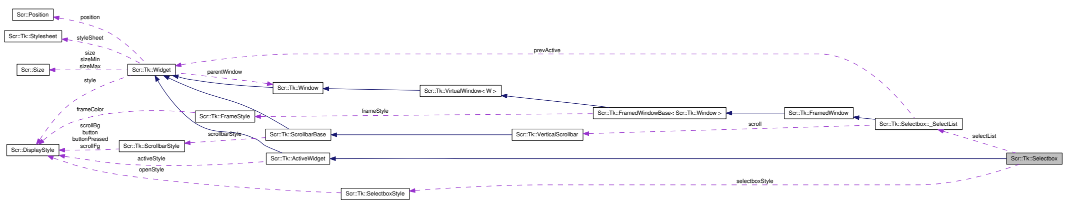
6.40 Scr::Tk::Selectbox Class Reference

`#include <selectbox.h++>`

Inheritance diagram for Scr::Tk::Selectbox:



Collaboration diagram for Scr::Tk::Selectbox:



Public Member Functions

- Selectbox (Uint width, const DisplayStyle &_style=SELECTBOX_DEFAULT_STYLE, const DisplayStyle &_activeStyle=SELECTBOX_DEFAULT_ACTIVESTYLE, const SelectboxStyle &_selectboxStyle=SelectboxStyle()) throw ()
- Uint AddOption (const std::string &name) throw ()
- const std::string & GetOption () const throw (NoSuchOption)
- void DelOption (Uint id) throw (NoSuchOption)
- void OnRedraw (Screen &screen) throw ()
- void OnFocus (FocusPolicy focusPolicy) throw ()
- void OnUnFocus (FocusPolicy focusPolicy) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Attributes

- SelectboxStyle selectboxStyle

  internal style

- _SelectList selectList

  list of options

- bool opened

  indicated whether the list of options is open

Classes

- class _SelectList

  Actual list of available options at Selectbox.

6.40.1 Detailed Description

Widget allowing to select one of available options.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 Selectbox::Selectbox (Uint *width*, const DisplayStyle & *_style* = `SELECTBOX_DEFAULT_STYLE`, const DisplayStyle & *_activeStyle* = `SELECTBOX_DEFAULT_ACTIVESTYLE`, const SelectboxStyle & *_selectboxStyle* = SelectboxStyle `()`) throw ()

Parameters:

width

_style

_activeStyle

_selectboxStyle

6.40.3 Member Function Documentation

6.40.3.1 Uint Selectbox::AddOption (const std::string & *name*) throw ()

Parameters:

name

Returns:

unique identifier

Adds new option to the list.

6.40.3.2 const std::string & Selectbox::GetOption () const throw (NoSuchOption)

Parameters:

id

Returns:

Selected option

Exceptions:

NoSuchOption if no option is selected

6.40.3.3 void Scr::Tk::Selectbox::DelOption (Uint *id*) throw (NoSuchOption) `[inline]`

Parameters:

id identifier of option to delete Deletes option from the list.

6.40.3.4 void Selectbox::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Widget.

6.40.3.5 void Selectbox::OnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element focused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::ActiveWidget.

6.40.3.6 void Selectbox::OnUnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element unfocused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::ActiveWidget.

6.40.3.7 virtual bool Scr::Tk::Selectbox::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::ActiveWidget.

6.40.3.8 virtual const char∗ Scr::Tk::Selectbox::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::ActiveWidget.

6.40.3.9 virtual const char∗ Scr::Tk::Selectbox::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::ActiveWidget.

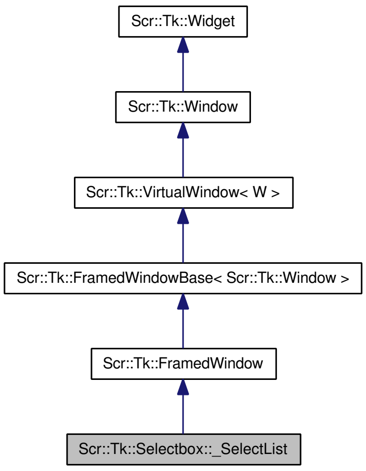
The documentation for this class was generated from the following files:

- include/rexio/tk/selectbox.h++
- lib/toolkit/src/selectbox.c++

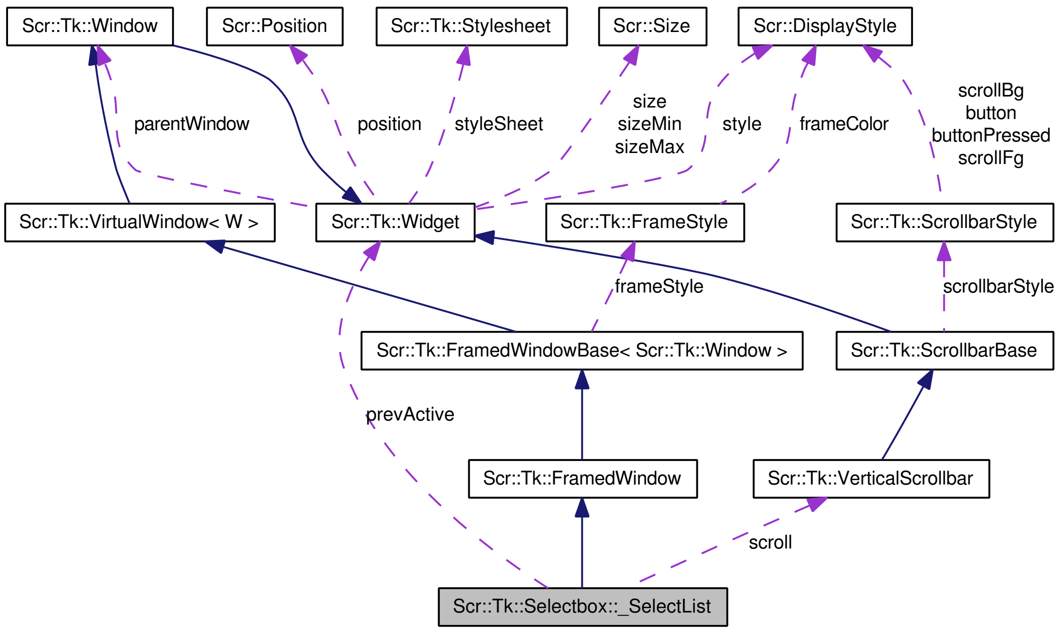
6.41 Scr::Tk::Selectbox::_SelectList Class Reference

Actual list of available options at Selectbox.

Inheritance diagram for Scr::Tk::Selectbox::_SelectList:


Scr::Tk::Widget
Scr::Tk::Window
Scr::Tk::VirtualWindow< W >
Scr::Tk::FramedWindowBase< Scr::Tk::Window >
Scr::Tk::FramedWindow
Scr::Tk::Selectbox::_SelectList

Collaboration diagram for Scr::Tk::Selectbox::_SelectList:



Public Member Functions

- void OnResize () throw ()
- void OnKeyDown (Key key) throw ()
- void OnFocus (FocusPolicy focustype) throw ()
- void OnUnFocus (FocusPolicy focustype) throw ()

Public Attributes

- VerticalScrollbar scroll

Scrollbar.

- Widget ∗ prevActive

6.41.1 Detailed Description

Actual list of available options at Selectbox.

6.41.2 Member Function Documentation

6.41.2.1 void Selectbox::_SelectList::OnResize () throw () `[virtual]`

Resize event. Do something i.e. adjust content to the new size. *VirtualWindow* specific: Has to be overloaded in deriving classes to handle proper resizing of containing window.

Reimplemented from Scr::Tk::FramedWindowBase< W >.

6.41.2.2 void Selectbox::_SelectList::OnKeyDown (Key *key*) throw () `[virtual]`

Parameters:

key keycode

Keyboard button press event.

Reimplemented from Scr::Tk::Window.

6.41.2.3 void Selectbox::_SelectList::OnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element focused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Window.

6.41.2.4 void Selectbox::_SelectList::OnUnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element unfocused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Window.

6.41.3 Member Data Documentation

6.41.3.1 Widget∗ Scr::Tk::Selectbox::_SelectList::prevActive

previous active widget at RootWindow to which the focus will have to be returned.

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

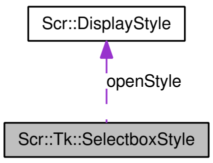
- include/rexio/tk/selectbox.h++
- lib/toolkit/src/selectbox.c++

6.42 Scr::Tk::SelectboxStyle Struct Reference

Selectbox specific style.

`#include <selectbox.h++>`

Collaboration diagram for Scr::Tk::SelectboxStyle:



Public Member Functions

- SelectboxStyle (const wchar_t _openButton=_DEFAULT_SELECTBOX_OPENBUTTON, const DisplayStyle &_openStyle=_DEFAULT_SELECTBOX_OPENSTYLE) throw ()

6.42.1 Detailed Description

Selectbox specific style.

Describes the way a specific selectbox is drawn.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 Scr::Tk::SelectboxStyle::SelectboxStyle (const wchar_t *_openButton* = `_DEFAULT_SELECTBOX_OPENBUTTON`, const DisplayStyle & *_openStyle* = `_DEFAULT_SELECTBOX_OPENSTYLE`) throw () `[inline]`

Parameters:

_openButton symbol for drawing opening symbol

_openStyle color for drawing the opening symbol

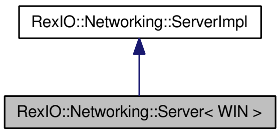
The documentation for this struct was generated from the following file:

- include/rexio/tk/selectbox.h++

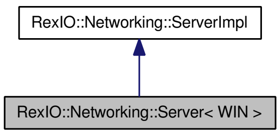
6.43 RexIO::Networking::Server< WIN > Class Template Reference

`#include <net.h++>`

Inheritance diagram for RexIO::Networking::Server< WIN >:


RexIO::Networking::ServerImpl
RexIO::Networking::Server< WIN >

Collaboration diagram for RexIO::Networking::Server< WIN >:


RexIO::Networking::ServerImpl
RexIO::Networking::Server< WIN >

6.43.1 Detailed Description

template<typename WIN> class RexIO::Networking::Server< WIN >

templatized version of ServerImpl (WIN parameter is class derivated from RootWindow

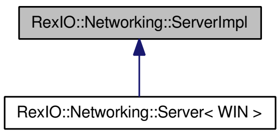
The documentation for this class was generated from the following file:

- include/rexio/net.h++

6.44 RexIO::Networking::ServerImpl Class Reference

`#include <net.h++>`

Inheritance diagram for RexIO::Networking::ServerImpl:


RexIO::Networking::ServerImpl
RexIO::Networking::Server< WIN >

Public Member Functions

- ServerImpl ()

  default constructor

- void Start (int portnum)
- void Stop ()

6.44.1 Detailed Description

Virtual base for server implementation has almost all code needed to operate as RexIO server. This class facilitates thread management, window creation and so on.

Note:

this class is not guaranteed to be thread safe. it uses some global data structures, and was not designed with many RexIO servers operated within one process, so please avoid id

6.44.2 Member Function Documentation

6.44.2.1 void ServerImpl::Start (int *portnum*)

start listening on specified port number

Parameters:

portnum	port number

6.44.2.2 void ServerImpl::Stop ()

end listening, send "terminate" messages to all clients. Then end.

Note:

this function is not guaranteed to succeed: if any thread is enters infinite loop, this function will wait until kill -9.

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

- include/rexio/net.h++
- lib/net/netconn.c++

6.45 Scr::Size Struct Reference

size container

```
#include <commons.h++>
```

Public Member Functions

- Size (Uint _height, Uint _width)

Public Attributes

- Uint height
- Uint width

6.45.1 Detailed Description

size container

6.45.2 Constructor & Destructor Documentation

6.45.2.1 Size::Size (Uint *_height*, Uint *_width*)

Parameters:

_height	height
_width	width

Simple constructor for convenient initialization and creation.

6.45.3 Member Data Documentation

6.45.3.1 Uint Scr::Size::height

height property

6.45.3.2 Uint Scr::Size::width

width property

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

- include/rexio/commons.h++
- lib/screen/src/core/commons.c++

6.46 Scr::Tk::Stylesheet Class Reference

CSS-like properties holder.

`#include <stylesheet.h++>`

Public Types

- enum PropertyType

  Type specifying Property value.

Public Member Functions

- const Property & GetProperty (const Widget &w, const std::string &property) throw (Properties::NoSuchProperty)
- void SetProperty (const std::string &className, const std::string &property, const Property &value) throw ()
- Stylesheet (std::istream &ss) throw (ParsingError, Screen::InvalidUTF8)

Private Types

- typedef std::map< std::string, Properties $*$ > ClassMap

  Type to bind class names to their properties.

Private Member Functions

- Property ParseValue (const std::string &valuestr) throw (BadValue, Screen::InvalidUTF8)

Private Attributes

- ClassMap classes

  Allows accessing properties of different classes.

Classes

- class Property

  Class holding multiple possible types of values.

6.46.1 Detailed Description

CSS-like properties holder.

Stylesheet is a class which can hold different properties for different classes. There are few value types supported. It incomporates complete parser.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 Stylesheet::Stylesheet (std::istream & *ss*) throw (ParsingError, Screen::InvalidUTF8)

Parameters:

ss stream of CSS-like formatted data

Parses the specified buffer for later access.

Exceptions:

ParsingError is thrown had the buffer was not properly formatted.

Screen::InvalidUTF8 is thrown if an UTF-8 character enclosed in single braces ' ' is not in correct UTF-8 format.

6.46.3 Member Function Documentation

6.46.3.1 Stylesheet::Property Stylesheet::ParseValue (const std::string & *valuestr*) throw (BadValue, Screen::InvalidUTF8) `[private]`

Parameters:

valuestr unparsed value string

Returns:

Property properly interpretted and converted valuestr

The function takes a crude string which is the following part of CSS syntax: **property: value**; and converts it into the internal value holder.

Exceptions:

BadValue is throws if the valuestr cannot be parsed.

Screen::InvalidUTF8 is thrown if an UTF-8 character enclosed in single braces ' ' is not in correct UTF-8 format.

6.46.3.2 const Stylesheet::Property & Stylesheet::GetProperty (const Widget & *w*, const std::string & *property*) throw (Properties::NoSuchProperty)

Parameters:

w widget to check

property

Returns:

reference to found property

Find certain property value for a widget.

Exceptions:

Properties::NoSuchProperty is thrown if no data has been found.

6.46.3.3 void Stylesheet::SetProperty (const std::string & *className*, const std::string & *property*, const Property & *value*) throw ()

Parameters:

className

property

value Bind a certain vlaue to certain class's property.

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

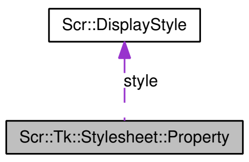
- include/rexio/tk/stylesheet.h++
- lib/toolkit/src/stylesheet.c++

6.47 Scr::Tk::Stylesheet::Property Class Reference

Class holding multiple possible types of values.

```
#include <stylesheet.h++>
```

Collaboration diagram for Scr::Tk::Stylesheet::Property:



Public Member Functions

- const Property & operator= (const Property &old)
- Property (const Property &old)
- Property (const DisplayStyle &_style)
- Property (wchar_t _symbol)
- Property (Uint32 _number)
- Property (const std::string &_str)
- PropertyType GetType () const throw ()
- operator DisplayStyle () const
- operator const std::string () const
- operator Uint32 () const
- operator wchar_t () const
- ∼Property ()

Private Attributes

- PropertyType type

  Current type.

6.47.1 Detailed Description

Class holding multiple possible types of values.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 Scr::Tk::Stylesheet::Property::Property (const Property & *old*) `[inline]`

Parameters:

old Copy constructor handling the allocated objects.

6.47.2.2 Scr::Tk::Stylesheet::Property::Property (const DisplayStyle & *_style*) `[inline]`

Parameters:

_style data to hold Specialized constructor for holding DisplayStyle data.

6.47.2.3 Scr::Tk::Stylesheet::Property::Property (wchar_t *_symbol*) `[inline]`

Parameters:

_symbol data to hold Specialized constructor for holding wchar_t data.

6.47.2.4 Scr::Tk::Stylesheet::Property::Property (Uint32 *_number*) `[inline]`

Parameters:

_number data to hold Specialized constructor for holding Uint32 data.

6.47.2.5 Scr::Tk::Stylesheet::Property::Property (const std::string & *_str*) `[inline]`

Parameters:

_str data to hold Specialized constructor for holding std::string data.

6.47.2.6 Scr::Tk::Stylesheet::Property::∼Property () `[inline]`

Smart destructor, deleting type specific data.

6.47.3 Member Function Documentation

6.47.3.1 const Property& Scr::Tk::Stylesheet::Property::operator= (const Property & *old*) `[inline]`

Parameters:

old Assign operator handling the allocated objects.

6.47.3.2 PropertyType Scr::Tk::Stylesheet::Property::GetType () const throw () `[inline]`

Returns:

type of a Property

6.47.3.3 Scr::Tk::Stylesheet::Property::operator DisplayStyle () const `[inline]`

Autoconversion to DisplayStyle.

6.47.3.4 Scr::Tk::Stylesheet::Property::operator const std::string () const `[inline]`

Autoconversion to std::string..

6.47.3.5 Scr::Tk::Stylesheet::Property::operator Uint32 () const `[inline]`

Autoconversion to Uint32.

6.47.3.6 Scr::Tk::Stylesheet::Property::operator wchar_t () const `[inline]`

Autoconversion to wchar_t.

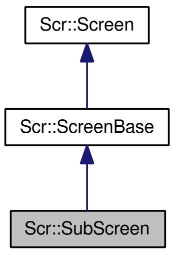
The documentation for this class was generated from the following file:

- include/rexio/tk/stylesheet.h++

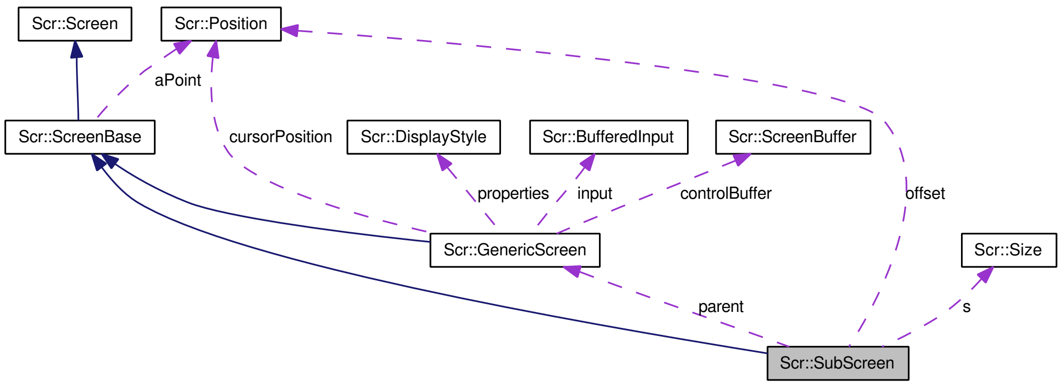
6.48 Scr::SubScreen Class Reference

```
#include <subscreen.h++>
```

Inheritance diagram for Scr::SubScreen:



Collaboration diagram for Scr::SubScreen:



Public Member Functions

- SubScreen (GenericScreen &_parent, Uint _y_offset, Uint _x_offset, Uint _h, Uint _w) throw ()
- virtual void Clear () throw ()
- virtual void SetBgColor (Bg::Color col) throw ()
- virtual void SetFgColor (Fg::Color col) throw ()
- virtual void SetFgStyle (Fg::Style s) throw ()
- virtual void GotoYX (Uint y, Uint x) throw (GotoOutOfRange)
- virtual void AddText (const char ∗text) throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const std::string &text) throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const wchar_t ∗text) throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddText (const std::wstring &text) throw (PrintOutOfRange, IllegalCharacter)
- virtual Uint AddTextCols (const wchar_t ∗text, Uint limitcols) throw (PrintOutOfRange, IllegalCharacter)
- virtual Uint AddTextCols (const std::wstring &text, Uint limitcols) throw (PrintOutOfRange, IllegalCharacter)
- virtual void HorizontalLine (char c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void HorizontalLine (wchar_t c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void VerticalLine (char c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void VerticalLine (wchar_t c, Uint n) throw (PrintOutOfRange, IllegalCharacter)
- virtual void Rectangle (char c, const Size &s) throw (PrintOutOfRange, IllegalCharacter)
- virtual void Rectangle (wchar_t c, const Size &s) throw (PrintOutOfRange, IllegalCharacter)
- virtual void AddCharacter (char c) throw (PrintOutOfRange)

- virtual void AddCharacter (wchar_t c) throw (PrintOutOfRange, IllegalCharacter)
- virtual void ForceCursorPosition (Position p) throw (RangeError)
- virtual void HideCursor () throw (CursorVisibilityNotSupported)
- virtual void ShowCursor () throw (CursorVisibilityNotSupported)
- virtual void Refresh () throw (ConnectionError)
- virtual void Resize (Uint rows, Uint cols) throw (SubscreenResize)
- virtual const char ∗ GetType () const throw (TerminalTypeUnknown)
- virtual Uint GetHeight () const throw ()
- virtual Uint GetWidth () const throw ()
- virtual bool GetCursorVisibility () const throw ()
- virtual Screen ∗ CreateSubScreen (Uint _y_offset, Uint _x_offset, Uint _h, Uint _w) throw (SubscreenOutOfRange)

Protected Member Functions

- void ParentGotoYXForPrinting () throw (PrintOutOfRange)

Protected Attributes

- GenericScreen & parent
- Position offset
- Size s

6.48.1 Detailed Description

Subscreen may be considered a specified region of screen limited to one rectangle. Subscreen does not provide it's own buffer, so it can be used as range for specific procedure rather than a layer. It allows all actions, but limited to it's width and height. It is useful for implementing procedures drawing specific features, i.e. widgets in UI toolkit.

Strict range limitation is achieved by disabling of Scr::SubScreen::Resize member function

6.48.2 Constructor & Destructor Documentation

6.48.2.1 Scr::SubScreen::SubScreen (GenericScreen & *_parent*, Uint *_y_offset*, Uint *_x_offset*, Uint *_h*, Uint *_w*) throw ()

Parameters:

_parent reference to parent screen

_y_offset vertical distance from top of containing (parent) screen to top of this

_x_offset horizontal distance from left edge of containing (parent) screen to left edge of this

_h height

_w width

6.48.3 Member Function Documentation

6.48.3.1 void Scr::SubScreen::ParentGotoYXForPrinting () throw (PrintOutOfRange) `[inline, protected]`

Call GotoYX for parent. Rethrow possible exception as Printing exception.

6.48.3.2 void Scr::SubScreen::Clear () throw () `[virtual]`

Fills rectangle defined by this subscreen with current background color, directly on containing buffer (so it may be later hidden by containing buffer)

Implements Scr::Screen.

6.48.3.3 void Scr::SubScreen::SetBgColor (Bg::Color *col*) throw () `[virtual]`

Parameters:

col color

Subscreen does not have it's own DisplayProperties, so it calls SetBgColor for parent screen

Implements Scr::Screen.

6.48.3.4 void Scr::SubScreen::SetFgColor (Fg::Color *col*) throw () `[virtual]`

Parameters:

col color

Subscreen does not have it's own DisplayProperties, so it calls SetFgColor for parent screen

Implements Scr::Screen.

6.48.3.5 void Scr::SubScreen::SetFgStyle (Fg::Style *s*) throw () `[virtual]`

Parameters:

s style

Subscreen does not have it's own DisplayProperties, so it calls SetFgStyle for parent screen

Implements Scr::Screen.

6.48.3.6 void Scr::SubScreen::GotoYX (Uint *y*, Uint *x*) throw (GotoOutOfRange) `[virtual]`

Parameters:

x

y this does not access directly to parent window, as SubScreen has it's own YX coordinates

Implements Scr::Screen.

6.48.3.7 void Scr::SubScreen::AddText (const char * *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text Print text directly on parent buffer

Note:

it means, that first appropriate GotoYX must be called for parent, so it modifies not only contents of buffer, but also coordinates of its active point.

Implements Scr::Screen.

6.48.3.8 void Scr::SubScreen::AddText (const std::string & *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text Same as above.

Implements Scr::Screen.

6.48.3.9 void Scr::SubScreen::AddText (const wchar_t $*$ *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text

Same as above

Implements Scr::Screen.

6.48.3.10 void Scr::SubScreen::AddText (const std::wstring & *text*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text

Same as above, but UNICODE

Implements Scr::Screen.

6.48.3.11 Uint Scr::SubScreen::AddTextCols (const wchar_t $*$ *text*, Uint *limitcols*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text wide string

limitcols max width in columns

Function prints AT MOST limitcols wide string. Width means number of columns, which is not the same thing as number of characters, as most CJK glyphs are multicolumn.

Exceptions:

PrintOutOfRange is thrown if initial position of active point is invalid, or if text is too long (as function does not support line breaks).

IllegalCharacter will be thrown if text supplied is not a valid UTF-8 string (even "overlong sequences" will be considered illegal (according to an apropriate RFC

See also:

Screen::AddText(const char ∗ text) for extensive description

Implements Scr::Screen.

6.48.3.12 Uint SubScreen::AddTextCols (const std::wstring & *text*, Uint *limitcols*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

text wide string

limitcols max width in columns

Function prints AT MOST limitcols wide string. Width means number of columns, which is not the same thing as number of characters, as most CJK glyphs are multicolumn.

Implements Scr::Screen.

6.48.3.13 void Scr::SubScreen::HorizontalLine (char *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c ASCII character

n number of repetitions (length of line)

Function adds horizontal line of n characters c.

Implements Scr::Screen.

6.48.3.14 void Scr::SubScreen::HorizontalLine (wchar_t *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c UNICODE character

n number of repetitions (length of line)

Function adds horizontal line of n characters c.

Implements Scr::Screen.

6.48.3.15 void Scr::SubScreen::VerticalLine (char *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c ASCII character

n number of repetitions (length of line)

Function adds verticel line of n characters c.

Implements Scr::Screen.

6.48.3.16 void Scr::SubScreen::VerticalLine (wchar_t *c*, Uint *n*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c UNICODE character

n number of repetitions (length of line)

Function adds vertical line of n characters c.

Implements Scr::Screen.

6.48.3.17 void Scr::SubScreen::Rectangle (char *c*, const Size & *s*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c character used to create rectangle

s dimensions of rectangle

Function creates rectangle of characters. s specifies number of rows and number of repetitions of character c in each row.

Implements Scr::Screen.

6.48.3.18 void Scr::SubScreen::Rectangle (wchar_t *c*, const Size & *s*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c character used to create rectangle

s dimensions of rectangle

Function creates rectangle of characters. s specifies number of rows and number of repetitions of character c in each row.

Implements Scr::Screen.

6.48.3.19 void Scr::SubScreen::AddCharacter (char *c*) throw (PrintOutOfRange) `[virtual]`

Parameters:

c Print character directly on parent buffer

Note:

as for AddText, it modifies not only contents of buffer, but also coordinates of its active point.

Implements Scr::Screen.

6.48.3.20 void Scr::SubScreen::AddCharacter (wchar_t *c*) throw (PrintOutOfRange, IllegalCharacter) `[virtual]`

Parameters:

c Print UNICODE character directly on parent buffer

Note:

as for AddText, it modifies not only contents of buffer, but also coordinates of its active point.

Implements Scr::Screen.

6.48.3.21 void Scr::SubScreen::ForceCursorPosition (Position *p*) throw (RangeError) `[virtual]`

Parameters:

p position

mapped to parent

Implements Scr::Screen.

6.48.3.22 void Scr::SubScreen::HideCursor () throw (CursorVisibilityNotSupported) `[virtual]`

make cursor invisible

Implements Scr::Screen.

6.48.3.23 void Scr::SubScreen::ShowCursor () throw (CursorVisibilityNotSupported) `[virtual]`

make it visible again

Implements Scr::Screen.

6.48.3.24 void Scr::SubScreen::Refresh () throw (ConnectionError) `[virtual]`

force refresh of parent buffer

Implements Scr::Screen.

6.48.3.25 void Scr::SubScreen::Resize (Uint *rows*, Uint *cols*) throw (SubscreenResize) `[virtual]`

Parameters:

rows

cols

Exceptions:

Scr::Screen::SubscreenResize is thrown always, as subscreen can not be resized

Implements Scr::Screen.

6.48.3.26 const char ∗ Scr::SubScreen::GetType () const throw (TerminalTypeUnknown) `[virtual]`

Return type of parent screen type (effectively it is the type of underlying real screen)

Implements Scr::Screen.

6.48.3.27 Scr::Uint Scr::SubScreen::GetHeight () const throw () `[virtual]`

Returns:

Current height of screen

Implements Scr::Screen.

6.48.3.28 Scr::Uint Scr::SubScreen::GetWidth () const throw () `[virtual]`

Returns:

Current width of screen

Implements Scr::Screen.

6.48.3.29 bool Scr::SubScreen::GetCursorVisibility () const throw () `[virtual]`

Returns:

true if cursor is visible, false if it ishidden

See also:

ShowCursor HideCursor

Implements Scr::Screen.

6.48.3.30 Scr::Screen ∗ Scr::SubScreen::CreateSubScreen (Uint *_y_offset*, Uint *_x_offset*, Uint *_h*, Uint *_w*) throw (SubscreenOutOfRange) `[virtual]`

Parameters:

_y_offset vertical offset from top edge of this screen to top edge of new SubScreen.
_x_offset horizontal offser
_h height
_w with

Returns:

pointer to new SubScreen (programmer will have to free it's resources to prevent memory leak and other errors).

Exceptions:

Scr::Screen::SubscreenOutOfRange is thrown when too big subscreen requested or inapropriate position specified

Implements Scr::Screen.

6.48.4 Member Data Documentation

6.48.4.1 GenericScreen& Scr::SubScreen::parent `[protected]`

reference to parent screen

6.48.4.2 Position Scr::SubScreen::offset `[protected]`

vertical distance from top of containing (parent) screen to top of this and horizontal distance from its left edge.

6.48.4.3 Size Scr::SubScreen::s `[protected]`

Width and height of screen

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

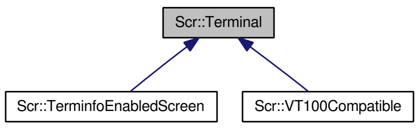
- lib/screen/include/subscreen.h++
- lib/screen/src/subscreen/subscreen.c++

6.49 Scr::Terminal Class Reference

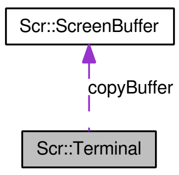
base class containing data fields typical to any terminal output type

`#include <terminal.h++>`

Inheritance diagram for Scr::Terminal:



Collaboration diagram for Scr::Terminal:



Protected Attributes

- struct {
  Uint x
    column
  Uint y
    row
  } termCoords

- ScreenBuffer copyBuffer

6.49.1 Detailed Description

base class containing data fields typical to any terminal output type

6.49.2 Member Data Documentation

6.49.2.1 struct { ... } Scr::Terminal::termCoords `[protected]`

Coordinates of cursor onscreen

6.49.2.2 ScreenBuffer Scr::Terminal::copyBuffer `[protected]`

Copy of expected screen contents - used to optimise Refresh() for transfer

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

- lib/screen/include/terminal.h++
- lib/screen/src/real/terminal.c++

6.50 Scr::TI::TerminfoCore Class Reference

Terminfo subsystem core: manages entries etc.

`#include <terminfo.h++>`

Public Member Functions

- void CleanUp () throw ()

Static Public Member Functions

- static void Initialize () throw (FailedToOpenDatabase)
- static bool GetDatabaseStatus () throw (DatabaseNotOpen)
- static const TerminfoEntry & GetTerminfo (const char ∗name) throw (NotSupportedTerminal-Type,FailedToOpenDatabase)
- static void FreeTerminfoEntry () throw ()

Private Member Functions

- TerminfoCore () throw ()
- ∼TerminfoCore () throw ()
- const TerminfoEntry & __GetTerminfo (const char ∗name) throw (NotSupportedTerminalType)

6.50.1 Detailed Description

Terminfo subsystem core: manages entries etc.

As this class is a singleton class, only one it’s instance may exist in the same time. don’t bother calling it’s constructor manually, as this will result in exiting program at all.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 TerminfoCore::TerminfoCore () throw () `[private]`

Default constructor; called by static GetTerminfo

Exceptions:

Scr::TI::FailedToOpenDatabase is thrown when no database found in supported format.

6.50.2.2 TerminfoCore::∼TerminfoCore () throw () `[private]`

Default destructor

6.50.3 Member Function Documentation

6.50.3.1 const TerminfoEntry & TerminfoCore::__GetTerminfo (const char ∗ *name*) throw (NotSupportedTerminalType) `[private]`

Function returns reference to TerminfoEntry object. If it was already retrieved, reference to existing one is returned. Otherwise new is created.

Parameters:

name name of terminal type ($TERM)

6.50.3.2 void TerminfoCore::Initialize () throw (FailedToOpenDatabase) `[static]`

This function forces initialization of terminfo database subsystem

6.50.3.3 bool TerminfoCore::GetDatabaseStatus () throw (DatabaseNotOpen) `[static]`

Returns:

true if database was successfully opened

6.50.3.4 const TerminfoEntry & TerminfoCore::GetTerminfo (const char ∗ *name*) throw (NotSupportedTerminalType,FailedToOpenDatabase) `[static]`

Parameters:

name $TERM

Returns:

const reference to terminfo entry object

Exceptions:

Scr::TI::NotSupportedTerminalType is thrown when not supported terminal type is requested

Scr::TI::FailedToOpenDatabase is thrown when no database found in supported format.

6.50.3.5 void TerminfoCore::CleanUp () throw ()

Force destruction of terminfo subsystem. This may cause numerous problem while any objects are still referencing terminfo entries. This function frees all TI resources if any allocated. Otherwise it won't do anything (so that there is no rish of "double free error").

6.50.3.6 void TerminfoCore::FreeTerminfoEntry () throw () `[static]`

Function conditionally cleans up terminfo connectivity subsystem.

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

- lib/screen/include/terminfo.h++
- lib/screen/src/terminfo/terminfocore.c++

6.51 Scr::TI::TerminfoDatabase Class Reference

terminfo database finds system database and fetches entries

`#include <terminfodatabase.h++>`

Public Member Functions

- TerminfoDatabase () throw ()
- boost::shared_ptr< std::ifstream > OpenFile (const char ∗name) throw (FailedToOpenDatabase, NotSupportedTerminalType, FailedToLoadDatabaseEntry)
- bool GetDatabaseStatus () throw ()

6.51.1 Detailed Description

terminfo database finds system database and fetches entries

6.51.2 Constructor & Destructor Documentation

6.51.2.1 TerminfoDatabase::TerminfoDatabase () throw ()

Default constructor looks for terminfo resources

6.51.3 Member Function Documentation

6.51.3.1 boost::shared_ptr< std::ifstream > TerminfoDatabase::OpenFile (const char ∗ *name*) throw (FailedToOpenDatabase, NotSupportedTerminalType, FailedToLoadDatabaseEntry)

Parameters:

name $TERM

Returns:

binary file containing term info.

6.51.3.2 bool TerminfoDatabase::GetDatabaseStatus () throw ()

Returns:

true if database was successfully opened

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

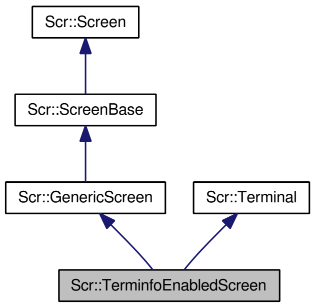
- lib/screen/src/terminfo/terminfodatabase.h++
- lib/screen/src/terminfo/terminfodatabase.c++

6.52 Scr::TerminfoEnabledScreen Class Reference

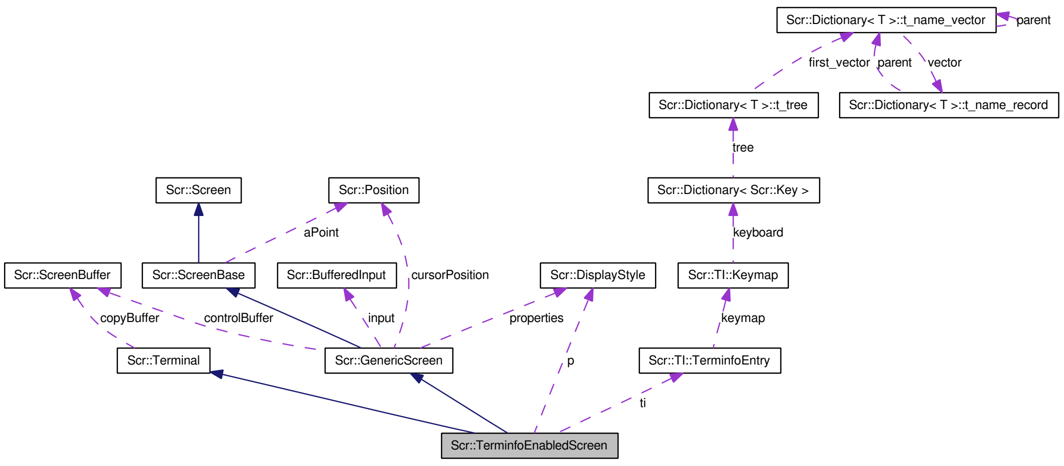
class representing terminal controlled according to terminfo database

`#include <terminfoenabled.h++>`

Inheritance diagram for Scr::TerminfoEnabledScreen:



Collaboration diagram for Scr::TerminfoEnabledScreen:



Public Member Functions

- virtual void Refresh () throw (ConnectionError)
- virtual void Resize (Uint rows, Uint cols) throw ()
- virtual void CleanUp () throw (ConnectionError)

Protected Member Functions

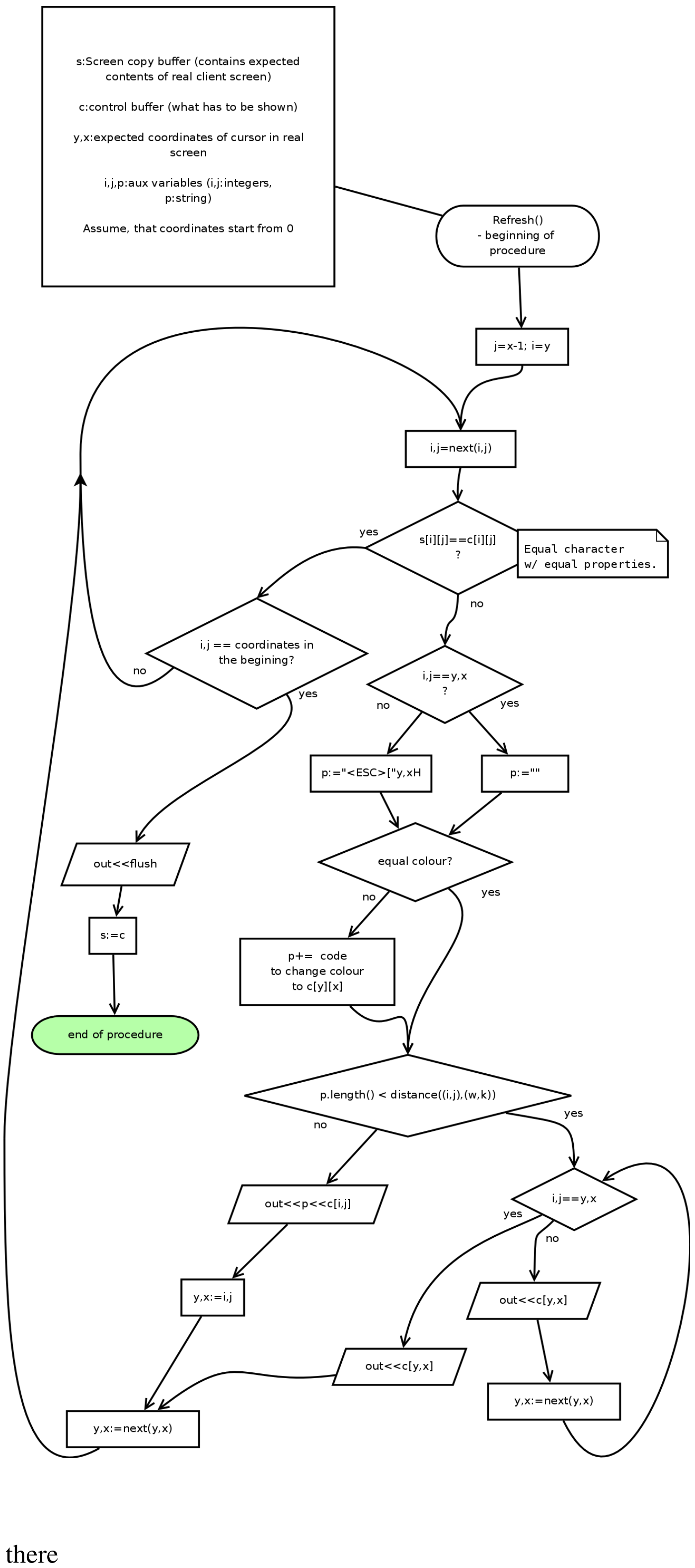
- virtual Key DecodeKeyPressed () throw (Connection::UnsupportedKey,Screen::InvalidUTF8)

6.52.1 Detailed Description

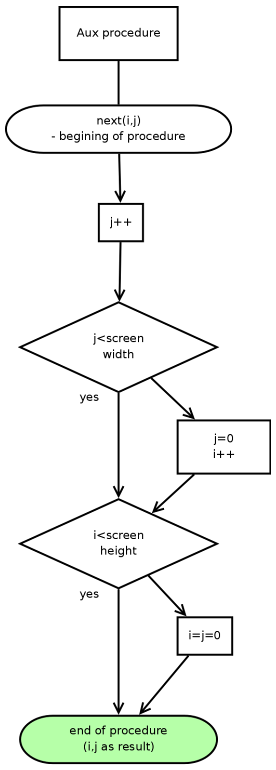
class representing terminal controlled according to terminfo database

This class provides full implementation of Scr::Screen abstract interface in terms of capabilities of any terminal described in terminfo database.

Algorithm for Refresh()


s:Screen copy buffer (contains expected contents of real client screen)
c:control buffer (what has to be shown)
y,x:expected coordinates of cursor in real screen
i,j,p:aux variables (i,j:integers, p:string)
Assume, that coordinates start from 0
Refresh() - beginning of procedure
j=x-1; i=y
i,j=next(i,j)
s[i][j]==c[i][j] ?
Equal character w/ equal properties.
yes
no
i,j == coordinates in the begining?
no
yes
i,j==y,x ?
no
yes
p:="<ESC>["y,xH
p:=""
out<<flush
s:=c
end of procedure
equal colour?
no
yes
p+= code to change colour to c[y][x]
p.length() < distance((i,j),(w,k))
no
yes
out<<p<<c[i,j]
y,x:=i,j
i,j==y,x
yes
no
out<<c[y,x]
out<<c[y,x]
y,x:=next(y,x)
y,x:=next(y,x)

Aux procedure used there



6.52.2 Member Function Documentation

6.52.2.1 Scr::Key Scr::TerminfoEnabledScreen::DecodeKeyPressed () throw (Connection::UnsupportedKey,Screen::InvalidUTF8) `[protected, virtual]`

Minimum implementation supportingonly 12 basic functionkeys, arrows and few special, in several formats of VT100-like terminal emulators.

Reimplemented from Scr::GenericScreen.

6.52.2.2 void Scr::TerminfoEnabledScreen::Refresh () throw (ConnectionError) `[virtual]`

Full support for colour and refreshing algorithm optimized for transfer

Reimplemented from Scr::GenericScreen.

6.52.2.3 void Scr::TerminfoEnabledScreen::Resize (Uint *rows*, Uint *cols*) throw () `[virtual]`

Parameters:

rows

cols differs from Scr::GenericScreen::Resize only by the fact, that it supports copyBuffer

Reimplemented from Scr::GenericScreen.

6.52.2.4 void Scr::TerminfoEnabledScreen::CleanUp () throw (ConnectionError) `[virtual]`

Cleans screen up: restore default colours and clear (it is good to use this function while finishing application etc.)

Reimplemented from Scr::GenericScreen.

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

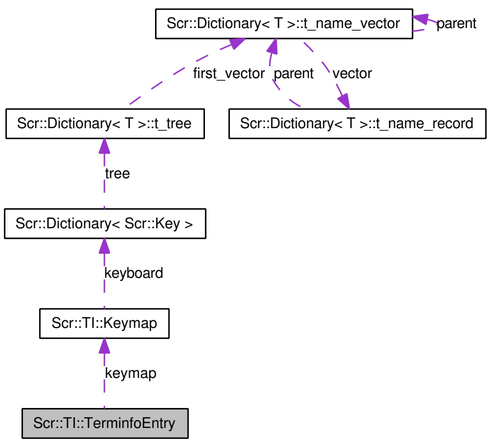
- lib/screen/include/terminfoenabled.h++
- lib/screen/src/real/terminfoenabled.c++

6.53 Scr::TI::TerminfoEntry Class Reference

Terminfo entry for single terminal type.

`#include <terminfo.h++>`

Collaboration diagram for Scr::TI::TerminfoEntry:


Scr::Dictionary< T >::t_name_vector
parent
first_vector
parent
vector
Scr::Dictionary< T >::t_tree
Scr::Dictionary< T >::t_name_record
tree
Scr::Dictionary< Scr::Key >
keyboard
Scr::TI::Keymap
keymap
Scr::TI::TerminfoEntry

Public Member Functions

- const std::string GotoYX (const Scr::Position &newPosition) const throw (CapabilityNotSupported)
- const std::string GotoYX (const Scr::Position &newPosition, const Scr::Position &oldPosition) const throw (CapabilityNotSupported)
- const std::string SetDisplayStyle (const Scr::DisplayStyle s) const throw (CapabilityNotSupported)
- const std::string SetDisplayStyle (const Scr::DisplayStyle newStyle, const Scr::DisplayStyle oldStyle) const throw (CapabilityNotSupported)
- const std::string ShowCursor () const throw (CapabilityNotSupported)
- const std::string HideCursor () const throw (CapabilityNotSupported)
- const std::string CursorHome () const throw (CapabilityNotSupported)

Protected Member Functions

- TerminfoEntry (std::ifstream &ifile) throw ()
- bool GetBoolean (int i) const throw ()
- short GetInteger (int i) const throw ()
- const char ∗ GetString (int i) const throw ()
- std::string ParseString (int i, Uint ∗param) const throw (CapabilityNotSupported,ParseError)

6.53.1 Detailed Description

Terminfo entry for single terminal type.

Terminfo entries will be read from system terminfo database (hashed database or hierarchical directory tree). Only way to obtain this class object is to call apropriate function of TerminfoCore object;

6.53.2 Constructor & Destructor Documentation

6.53.2.1 TerminfoEntry::TerminfoEntry (std::ifstream & *ifile*) throw () `[explicit, protected]`

Parameters:

ifile - resource reference to compiled terminfo file, that will be used to initialize this entry

Default constructor opens the file and reads all information in it.

6.53.3 Member Function Documentation

6.53.3.1 bool TerminfoEntry::GetBoolean (int *i*) const throw () `[protected]`

Parameters:

i cap. ID (enumerated in capabilities.h++)

Returns:

i'th boolean value

6.53.3.2 short TerminfoEntry::GetInteger (int *i*) const throw () `[protected]`

Parameters:

i cap. ID (enumerated in capabilities.h++)

Returns:

positive integer if feature is supported; -1 otherwise.

6.53.3.3 const char $*$ TerminfoEntry::GetString (int *i*) const throw () `[protected]`

Parameters:

i cap. ID (enumerated in capabilities.h++)

Returns:

c-style string if feature is supported. NULL pointer otherwise.

6.53.3.4 std::string TerminfoEntry::ParseString (int *i*, Uint ∗ *param*) const throw (CapabilityNotSupported,ParseError) `[protected]`

Parameters:

i cap. ID (enumerated in capabilities.h++)

param parameters (refer to terminfo(5) for parameter descriptions)

Parse parametrized string

Note:

implementation currently does not fully conform specification, however it does what is needed for this library.

6.53.3.5 const std::string TerminfoEntry::GotoYX (const Scr::Position & *newPosition*) const throw (CapabilityNotSupported)

Parameters:

newPosition new position (0,0 .. height-1,width-1)

Returns:

control string to set cursor position specific to this terminal type

Explicitly move cursor to the new position

6.53.3.6 const std::string TerminfoEntry::GotoYX (const Scr::Position & *newPosition*, const Scr::Position & *oldPosition*) const throw (CapabilityNotSupported)

Parameters:

newPosition new position of cursor (0,0 .. height-1,width-1)

oldPosition current position

Returns:

optimal control string to set cursor position specific to this terminal type

Recommended way of setting cursor position. This function selects way of setting position, that consumes least possible number of bytes.

Note:

dest and then source: the same argument order as for C library functions.

6.53.3.7 const std::string TerminfoEntry::SetDisplayStyle (const Scr::DisplayStyle *s*) const throw (CapabilityNotSupported)

Parameters:

s display style to be set

Returns:

control string to set display style for text.

6.53.3.8 const std::string TerminfoEntry::SetDisplayStyle (const Scr::DisplayStyle *newStyle*, const Scr::DisplayStyle *oldStyle*) const throw (CapabilityNotSupported)

Parameters:

newStyle display style to be set

oldStyle current style

Returns:

control string to set display style for text.

if current style is known, it is highly recommended to use this function as it will set minimum required subset of style attributes

6.53.3.9 const std::string TerminfoEntry::ShowCursor () const throw (CapabilityNotSupported)

Make cursor visible

6.53.3.10 const std::string TerminfoEntry::HideCursor () const throw (CapabilityNotSupported)

Make cursor invisible

6.53.3.11 const std::string TerminfoEntry::CursorHome () const throw (CapabilityNotSupported)

Move cursor to the begining-of-the-screen position (the same effect as GotoYX(Position(0,0)), but possibly faster)

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

- lib/screen/include/terminfo.h++
- lib/screen/src/terminfo/terminfoentry.c++

6.54 Scr::Vector Struct Reference

vector container

```
#include <commons.h++>
```

Public Member Functions

- Vector (Sint _rows, Sint _cols)

Public Attributes

- Sint rows

    offset in rows

- Sint cols

    offset in columns

6.54.1 Detailed Description

vector container

6.54.2 Constructor & Destructor Documentation

6.54.2.1 Vector::Vector (Sint *_rows*, Sint *_cols*)

Parameters:

_rows	rows offset
_cols	cols offset

Simple constructor for convenient initialization and creation.

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

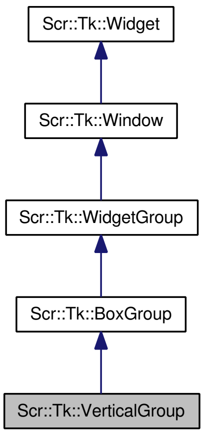
- include/rexio/commons.h++
- lib/screen/src/core/commons.c++

6.55 Scr::Tk::VerticalGroup Class Reference

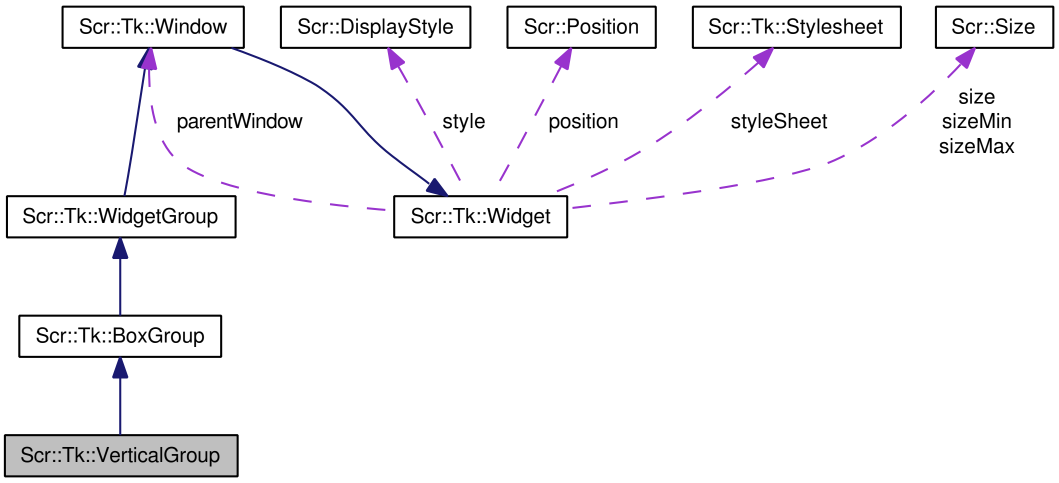
Vertical widget grouping capabilities.

`#include <verticalgroup.h++>`

Inheritance diagram for Scr::Tk::VerticalGroup:



Collaboration diagram for Scr::Tk::VerticalGroup:



Public Member Functions

- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Member Functions

- virtual void ArrangeContents () throw ()

6.55.1 Detailed Description

Vertical widget grouping capabilities.

Intelligently places the containing widgets among allocated space. Widgets are placed vertically.

6.55.2 Member Function Documentation

6.55.2.1 void VerticalGroup::ArrangeContents () throw () `[protected, virtual]`

where all magic is done :)

Implements Scr::Tk::BoxGroup.

6.55.2.2 virtual bool Scr::Tk::VerticalGroup::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::BoxGroup.

6.55.2.3 virtual const char∗ Scr::Tk::VerticalGroup::TypeName () const [inline, virtual]

Returns:

class name of this widget.

Reimplemented from Scr::Tk::BoxGroup.

6.55.2.4 virtual const char∗ Scr::Tk::VerticalGroup::ParentName () const [inline, virtual]

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::BoxGroup.

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

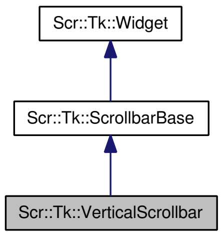
- include/rexio/tk/verticalgroup.h++
- lib/toolkit/src/verticalgroup.c++

6.56 Scr::Tk::VerticalScrollbar Class Reference

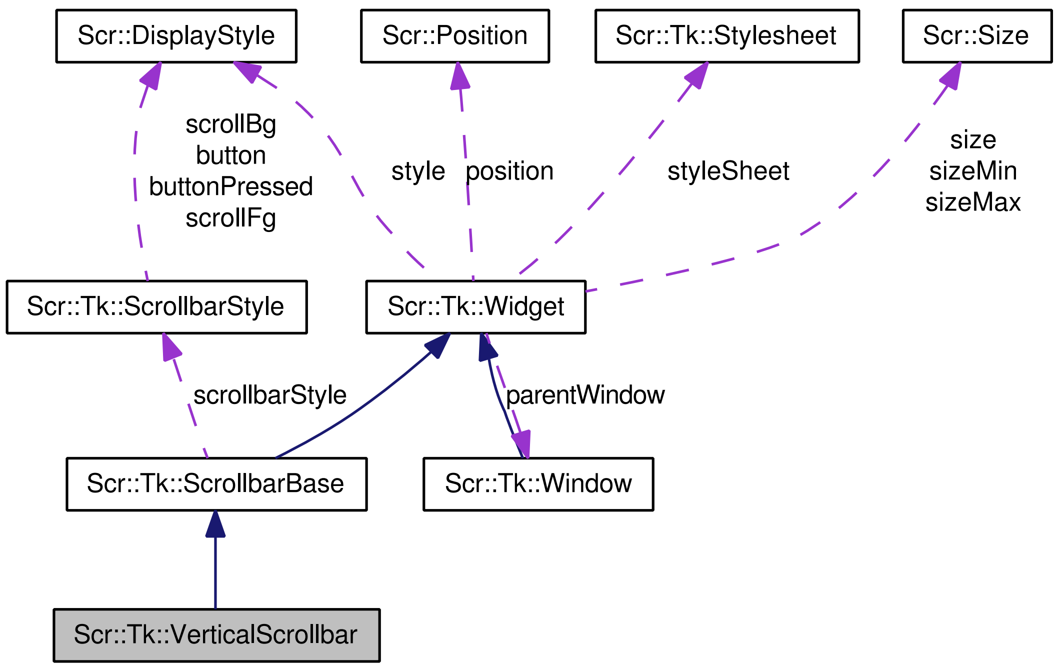
Vertical scrollbar.

`#include <scrollbar.h++>`

Inheritance diagram for Scr::Tk::VerticalScrollbar:



Collaboration diagram for Scr::Tk::VerticalScrollbar:



Public Member Functions

- VerticalScrollbar (Uint _height, const ScrollbarStyle &_scrollbarStyle=ScrollbarStyle()) throw ()
- virtual void OnRedraw (Screen &screen) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

6.56.1 Detailed Description

Vertical scrollbar.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 VerticalScrollbar::VerticalScrollbar (Uint *_height*, const ScrollbarStyle & *_scrollbarStyle* = ScrollbarStyle ( )) throw ()

Parameters:

_height

_scrollbarStyle

6.56.3 Member Function Documentation

6.56.3.1 void VerticalScrollbar::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

> ***screen*** reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

> the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Implements Scr::Tk::ScrollbarBase.

6.56.3.2 virtual bool Scr::Tk::VerticalScrollbar::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

> ***_className*** name of a class

Returns:

> whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::ScrollbarBase.

6.56.3.3 virtual const char* Scr::Tk::VerticalScrollbar::TypeName () const `[inline, virtual]`

Returns:

> class name of this widget.

Reimplemented from Scr::Tk::ScrollbarBase.

6.56.3.4 virtual const char* Scr::Tk::VerticalScrollbar::ParentName () const `[inline, virtual]`

Returns:

> parent class of this widget.

Reimplemented from Scr::Tk::ScrollbarBase.

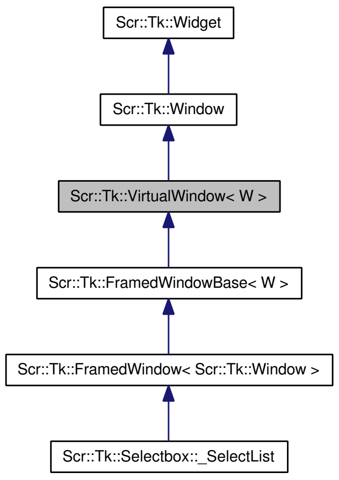
The documentation for this class was generated from the following files:

- include/rexio/tk/scrollbar.h++
- lib/toolkit/src/scrollbar.c++

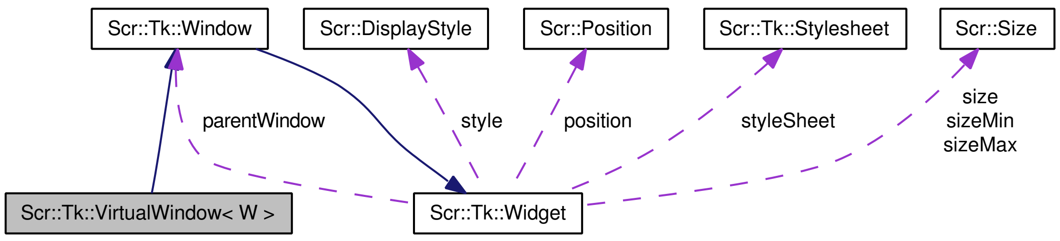
6.57 Scr::Tk::VirtualWindow< W > Class Template Reference

`#include <virtualwindow.h++>`

Inheritance diagram for Scr::Tk::VirtualWindow< W >:


Scr::Tk::Widget
Scr::Tk::Window
Scr::Tk::VirtualWindow< W >
Scr::Tk::FramedWindowBase< W >
Scr::Tk::FramedWindow< Scr::Tk::Window >
Scr::Tk::Selectbox::_SelectList

Collaboration diagram for Scr::Tk::VirtualWindow< W >:



Public Member Functions

- virtual void OnRedrawInside (Screen &screen) throw ()
- virtual void OnRedraw (Screen &screen) throw ()
- virtual void AddWidget (Widget &widget) throw ()
- virtual void DelWidget (Widget &widget) throw ()
- virtual void OnResize ()=0 throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Attributes

- W inside

    internal area, should have Window compatible interface.

6.57.1 Detailed Description

template<class W> class Scr::Tk::VirtualWindow< W >

Parameters:

W class of inside's window. Template for all framed windows. FramedWindowBase is basically a window having a separate internal window to which most of the calls(like AddWidget) are routed.

6.57.2 Member Function Documentation

6.57.2.1 template<class W> virtual void Scr::Tk::VirtualWindow< W >::OnRedrawInside (Screen & *screen*) throw () `[inline, virtual]`

Parameters:

screen cut-down to actual content area

Similiar to OnRedraw with an exception of providing cut-down screen.

6.57.2.2 template<class W> virtual void Scr::Tk::VirtualWindow< W >::OnRedraw (Screen & *screen*) throw () `[inline, virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Window.

Reimplemented in Scr::Tk::FramedWindowBase< W >, and Scr::Tk::FramedWindowBase< Scr::Tk::Window >.

6.57.2.3 template<class W> virtual void Scr::Tk::VirtualWindow< W >::AddWidget (Widget & *widget*) throw () `[inline, virtual]`

Parameters:

widget widget to attach to this window

Attach a widget to this window. Specifically, add it to the *elements*.

Exceptions:

ParentAlreadySet is thrown if the widget has already been attached to some other window.

WidgetAlreadyAdded if the widget is already attached to THIS window.

VirtualWindow specific: Passes the call to its internal window.

Reimplemented from Scr::Tk::Window.

6.57.2.4 template<class W> virtual void Scr::Tk::VirtualWindow< W >::DelWidget (Widget & *widget*) throw () `[inline, virtual]`

Parameters:

widget widget to detach from this window

Detach a widget from this window. Specifically, del it from the *elements*.

Exceptions:

WidgetNotPresent is thrown if the widget is not attached to this window.

VirtualWindow specific: Passes the call to its internal window.

Reimplemented from Scr::Tk::Window.

6.57.2.5 template<class W> virtual void Scr::Tk::VirtualWindow< W >::OnResize () throw () `[pure virtual]`

Resize event. Do something i.e. adjust content to the new size. *VirtualWindow* specific: Has to be overloaded in deriving classes to handle proper resizing of containing window.

Reimplemented from Scr::Tk::Window.

Implemented in Scr::Tk::FramedWindowBase< W >, Scr::Tk::Selectbox::_SelectList, and Scr::Tk::FramedWindowBase< Scr::Tk::Window >.

6.57.2.6 template<class W> virtual bool Scr::Tk::VirtualWindow< W >::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::Window.

Reimplemented in Scr::Tk::FramedWindow.

6.57.2.7 template<class W> virtual const char* Scr::Tk::VirtualWindow< W >::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::Window.

Reimplemented in Scr::Tk::FramedWindow.

6.57.2.8 template<class W> virtual const char∗ Scr::Tk::VirtualWindow< W >::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::Window.

Reimplemented in Scr::Tk::FramedWindow.

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

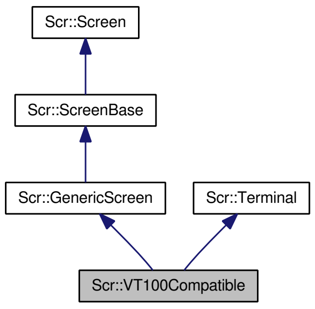
- include/rexio/tk/virtualwindow.h++

6.58 Scr::VT100Compatible Class Reference

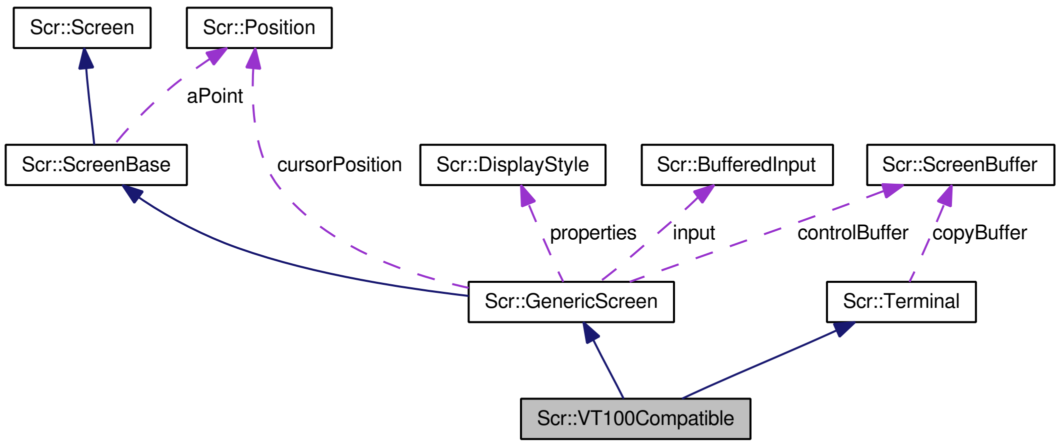
terminal compatible w/ DEC VT-100

`#include <vt100compatible.h++>`

Inheritance diagram for Scr::VT100Compatible:



Collaboration diagram for Scr::VT100Compatible:



Public Member Functions

- virtual void Refresh () throw (ConnectionError)

- virtual void Resize (Uint rows, Uint cols) throw ()
- virtual void CleanUp () throw (ConnectionError)

Protected Member Functions

- virtual Key DecodeKeyPressed () throw (Connection::UnsupportedKey,Screen::InvalidUTF8)

6.58.1 Detailed Description

terminal compatible w/ DEC VT-100

This class provides full implementation of Scr::Screen abstract interface in terms of capabilities of DEC VT100 compatible terminals. It will be used as fallback implementation when terminfo database is not availble

6.58.2 Member Function Documentation

6.58.2.1 Scr::Key Scr::VT100Compatible::DecodeKeyPressed () throw (Connection::UnsupportedKey,Screen::InvalidUTF8) `[protected, virtual]`

Minimum implementation supportingonly 12 basic functionkeys, arrows and few special, in several formats of VT100-like terminal emulators.

Reimplemented from Scr::GenericScreen.

6.58.2.2 void Scr::VT100Compatible::Refresh () throw (ConnectionError) `[virtual]`

Full support for colour and refreshing algorithm optimized for transfer

Reimplemented from Scr::GenericScreen.

6.58.2.3 void Scr::VT100Compatible::Resize (Uint *rows*, Uint *cols*) throw () `[virtual]`

Parameters:

rows

cols differs from Scr::GenericScreen::Resize only by the fact, that it supports copyBuffer

Reimplemented from Scr::GenericScreen.

6.58.2.4 void Scr::VT100Compatible::CleanUp () throw (ConnectionError) `[virtual]`

Cleans screen up: restore default colours and clear (it is good to use this function while finishing application etc.)

Reimplemented from Scr::GenericScreen.

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

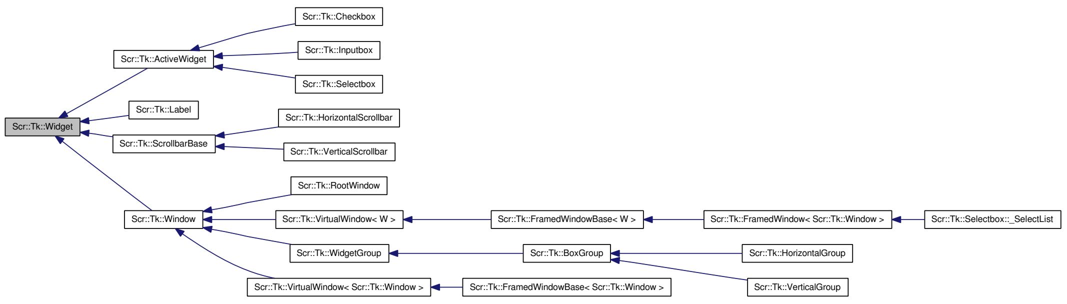
- lib/screen/include/vt100compatible.h++
- lib/screen/src/real/vt100compatible.c++

6.59 Scr::Tk::Widget Class Reference

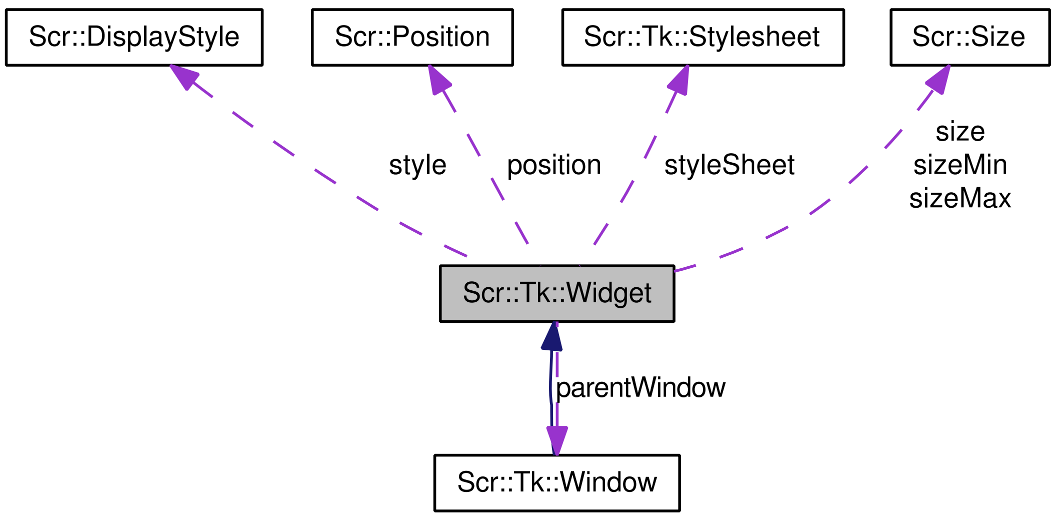
Base UI element.

`#include <widget.h++>`

Inheritance diagram for Scr::Tk::Widget:


Scr::Tk::Checkbox
Scr::Tk::ActiveWidget
Scr::Tk::Inputbox
Scr::Tk::Selectbox
Scr::Tk::Label
Scr::Tk::Widget
Scr::Tk::HorizontalScrollbar
Scr::Tk::ScrollbarBase
Scr::Tk::VerticalScrollbar
Scr::Tk::RootWindow
Scr::Tk::Window
Scr::Tk::VirtualWindow< W >
Scr::Tk::FramedWindowBase< W >
Scr::Tk::FramedWindow< Scr::Tk::Window >
Scr::Tk::Selectbox::_SelectList
Scr::Tk::WidgetGroup
Scr::Tk::BoxGroup
Scr::Tk::HorizontalGroup
Scr::Tk::VirtualWindow< Scr::Tk::Window >
Scr::Tk::FramedWindowBase< Scr::Tk::Window >
Scr::Tk::VerticalGroup

Collaboration diagram for Scr::Tk::Widget:



Public Types

- typedef std::vector< std::string > ClassHierarchy

Public Member Functions

- virtual void SetStylesheet (Stylesheet ∗_styleSheet) throw ()
- virtual void OnFocus (FocusPolicy focustype) throw ()
- virtual void OnUnFocus (FocusPolicy focustype) throw ()
- virtual void OnRedraw (Screen &screen) throw ()
- virtual void RedrawRequest () throw ()
- virtual void OnResize () throw ()
- virtual void OnKeyDown (Key key) throw ()

- virtual void OnExit () throw ()
- virtual void SetPosition (const Position &_pos) throw (ParentNotDefined)
- virtual void SetPosition (Uint _row, Uint _col) throw (ParentNotDefined)
- virtual Position GetPosition () const throw (ParentNotDefined)
- virtual void SetRow (Uint _row) throw (ParentNotDefined)
- virtual Uint GetRow () const throw (ParentNotDefined)
- virtual void SetCol (Uint _col) throw (ParentNotDefined)
- virtual Uint GetCol () const throw (ParentNotDefined)
- virtual void SetSize (const Size &_size) throw ()
- virtual void SetSize (Uint _height, Uint _width) throw ()
- virtual const Size & GetSize () const throw ()
- virtual void SetHeight (Uint _height) throw ()
- virtual Uint GetHeight () const throw ()
- virtual void SetWidth (Uint _width) throw ()
- virtual Uint GetWidth () const throw ()
- virtual void SetGeometry (const Position &_pos, const Size &_size) throw (ParentNotDefined)
- virtual void SetGeometry (Uint _row, Uint _col, Uint _height, Uint _width) throw (ParentNotDefined)
- virtual void SetMinSize (const Size &_size) throw ()
- virtual void SetMinSize (Uint _height, Uint _width) throw ()
- virtual const Size & GetMinSize () const throw ()
- virtual void SetMinHeight (Uint _height) throw ()
- virtual Uint GetMinHeight () const throw ()
- virtual void SetMinWidth (Uint _width) throw ()
- virtual Uint GetMinWidth () const throw ()
- virtual void SetMaxSize (const Size &_size) throw ()
- virtual void SetMaxSize (Uint _height, Uint _width) throw ()
- virtual const Size & GetMaxSize () const throw ()
- virtual void SetMaxHeight (Uint _height) throw ()
- virtual Uint GetMaxHeight () const throw ()
- virtual void SetMaxWidth (Uint _width) throw ()
- virtual Uint GetMaxWidth () const throw ()
- virtual void SetFocusPolicy (FocusPolicy _policy) throw ()
- virtual FocusPolicy GetFocusPolicy () const throw ()
- virtual void SetStyle (const DisplayStyle &style=DisplayStyle(Fg::System, Fg::Dark, Bg::System)) throw ()
- virtual const DisplayStyle & GetStyle () const throw ()
- void SetHidden (bool _hidden) throw ()
- bool IsHidden () const throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char * TypeName () const
- virtual const char * ParentName () const
- const ClassHierarchy & Hierarchy ()

Public Attributes

- std::string objectName

  Object name. Used for style targetting.

Protected Types

- enum FocusPolicy {

  NoFocus = 0x1, TabFocus = 0x1, ClickFocus = 0x2 , WheelFocus = WheelFocusUp|WheelFocusDown,

  StrongFocus = TabFocus|ClickFocus, AllFocus = TabFocus|ClickFocus|WheelFocus }

Protected Member Functions

- Widget (Uint _height, Uint _width, const DisplayStyle &_style=WIDGET_DEFAULT_STYLE) throw ()
- Widget (const DisplayStyle &_style=WIDGET_DEFAULT_STYLE) throw ()
- void SetParent (Window &window) throw (ParentAlreadySet)
- Window & GetParent () throw (ParentNotDefined)
- void RePartent (Window ∗window) throw ()

Protected Attributes

- Position position
- Size size
- Size sizeMax
- Size sizeMin
- DisplayStyle style
- bool hidden

Private Attributes

- Window ∗ parentWindow
- Stylesheet ∗ styleSheet

6.59.1 Detailed Description

Base UI element.

Widget - according to the dictionary, a device that is very useful for a particular job. In our case that can be any UI job and thus all the UI elements shall thereby be children of hit. Note that most widgets do not have their own buffer.

6.59.2 Member Typedef Documentation

6.59.2.1 std::vector< std::string > Scr::Tk::Widget::ClassHierarchy

Container holding the list of class names.

6.59.3 Member Enumeration Documentation

6.59.3.1 enum Scr::Tk::Widget::FocusPolicy `[protected]`

Focus policy defines a condition upon a widget can be focused.

Enumerator:

NoFocus	Nothing can focus.
TabFocus	Tabulator(or other switching key) can focus.
ClickFocus	Mouse click can focus.
WheelFocus	Mouse wheel can focus.
StrongFocus	TabFocus + Clickfocus.
AllFocus	Full service focus. :-).

6.59.4 Constructor & Destructor Documentation

6.59.4.1 Widget::Widget (Uint *_height*, Uint *_width*, const DisplayStyle & *_style* = `WIDGET_DEFAULT_STYLE`) throw () `[protected]`

This constructor should be used for widgets manually positioned. Widgets managed by *WidgetGroup* should be constructed with a more simple constructor.

Parameters:

_height	desired height
_width	desired width
_style	optional style

6.59.4.2 Widget::Widget (const DisplayStyle & *_style* = `WIDGET_DEFAULT_STYLE`) throw () `[protected]`

This constructor should be a preferred one if geometry and position of a Widget are to be managed by some *WidgetGroup*.

Parameters:

_style	optional style

6.59.5 Member Function Documentation

6.59.5.1 void Widget::SetParent (Window & *window*) throw (ParentAlreadySet) `[protected]`

Parameters:

window	parent of this widget

Parent of a widget can be set generally only once. After doing this, widget is ready to face the world so better prepare it properly first. This design decision has been made because of the constructor's primitive nature not being able to sustain all the possibilities.

See also:

ReParent

Exceptions:

ParentAlreadySet	is thrown had the parent already been set.

6.59.5.2 Window & Widget::GetParent () throw (ParentNotDefined) `[protected]`

Returns:

reference to parent window

Get reference to parent window.

Exceptions:

ParentNotSet is thrown if the parent window has not been yet specified.

6.59.5.3 void Widget::RePark (Window ∗ *window*) throw () `[protected]`

Parameters:

window pointer to parent of this widget, pass NULL after detaching the widget from window.

Provided for convenience. Sets the parent disregarding any conditions.

See also:

SetParent for general use.

6.59.5.4 void Widget::SetStylesheet (Stylesheet ∗ *_styleSheet*) throw () `[virtual]`

Parameters:

_styleSheet pointer to style data

Apply Stylesheet to this widget. Reinitialize any style properties if their alternatives are supplied.

Reimplemented in Scr::Tk::ActiveWidget, Scr::Tk::FramedWindowBase< W >, Scr::Tk::Inputbox, Scr::Tk::Label, Scr::Tk::ScrollbarBase, Scr::Tk::Window, and Scr::Tk::FramedWindowBase< Scr::Tk::Window >.

6.59.5.5 void Widget::OnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element focused. Only matters if a proper *focusPolicy* is set.

Reimplemented in Scr::Tk::ActiveWidget, Scr::Tk::Label, Scr::Tk::Selectbox::_SelectList, Scr::Tk::Selectbox, and Scr::Tk::Window.

6.59.5.6 void Widget::OnUnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element unfocused. Only matters if a proper *focusPolicy* is set.

Reimplemented in Scr::Tk::ActiveWidget, Scr::Tk::Label, Scr::Tk::Selectbox::_SelectList, Scr::Tk::Selectbox, and Scr::Tk::Window.

6.59.5.7 void Widget::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented in Scr::Tk::Checkbox, Scr::Tk::FramedWindowBase< W >, Scr::Tk::Inputbox, Scr::Tk::Label, Scr::Tk::RootWindow, Scr::Tk::ScrollbarBase, Scr::Tk::HorizontalScrollbar, Scr::Tk::VerticalScrollbar, Scr::Tk::Selectbox, Scr::Tk::VirtualWindow< W >, Scr::Tk::Window, Scr::Tk::FramedWindowBase< Scr::Tk::Window >, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.59.5.8 void Widget::RedrawRequest () throw () `[virtual]`

If the widget is attached to a window, it invokes parent's RedrawRequest with this widget. If it isn't attached, the function does nothing.

See also:

Window::RedrawRequest(Widget &w)

Reimplemented in Scr::Tk::Window.

6.59.5.9 void Widget::OnResize () throw () `[virtual]`

Resize event. Do something i.e. adjust content to the new size.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::FramedWindowBase< W >, Scr::Tk::Selectbox::_-SelectList, Scr::Tk::VirtualWindow< W >, Scr::Tk::Window, Scr::Tk::FramedWindowBase< Scr::Tk::Window >, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.59.5.10 void Widget::OnKeyDown (Key *key*) throw () `[virtual]`

Parameters:

key keycode

Keyboard button press event.

Reimplemented in Scr::Tk::ActiveWidget, Scr::Tk::Inputbox, Scr::Tk::Selectbox::_SelectList, and Scr::Tk::Window.

6.59.5.11 void Widget::OnExit () throw () `[virtual]`

Last event BEFORE the destructor call.

6.59.5.12 void Widget::SetPosition (const Position & *_pos*) throw (ParentNotDefined) `[virtual]`

Parameters:

_pos position new position

Set position of the Widget regarding to the *parentWindow*.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.13 void Widget::SetPosition (Uint *_row*, Uint *_col*) throw (ParentNotDefined) `[virtual]`

Parameters:

_row new row position

_col new column position

Set position of the Widget regarding to the *parentWindow*.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.14 Position Widget::GetPosition () const throw (ParentNotDefined) `[virtual]`

Returns:

position

Get position of the Widget regarding to the *parentWindow*.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.15 void Widget::SetRow (Uint *_row*) throw (ParentNotDefined) `[virtual]`

Parameters:

_row new row position

Set position of the Widget regarding to the *parentWindow* .

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.16 Uint Widget::GetRow () const throw (ParentNotDefined) `[virtual]`

Returns:

row position

Get position of the Widget regarding to the *parentWindow*.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.17 void Widget::SetCol (Uint *_col*) throw (ParentNotDefined) `[virtual]`

Parameters:

_col new column position

Set position of the Widget regarding to the *parentWindow*.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.18 Uint Widget::GetCol () const throw (ParentNotDefined) `[virtual]`

Returns:

col position

Get position of the Widget regarding to the *parentWindow*.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.19 void Widget::SetSize (const Size & *_size*) throw () `[virtual]`

Parameters:

_size new size

Set size of the Widget.

Note:

If entered size is bigger than *GetMaxSize()* or smaller than *GetMinSize()*, it will crop the entered value to the boundaries.

Reimplemented in Scr::Tk::Window.

6.59.5.20 void Widget::SetSize (Uint *_height*, Uint *_width*) throw () `[virtual]`

Parameters:

_height	new height
_width	new width

Set size of the Widget.

Note:

If entered size is bigger than *GetMaxSize()* or smaller than *GetMinSize()*, it will crop the entered value to the boundaries.

6.59.5.21 const Size & Widget::GetSize () const throw () `[virtual]`

Returns:

size

Get size of the Widget.

6.59.5.22 void Widget::SetHeight (Uint *_height*) throw () `[virtual]`

Parameters:

_height	new height

Set height of the Widget.

Note:

If entered size is bigger than *GetMaxSize()* or smaller than *GetMinSize()*, it will crop the entered value to the boundaries.

6.59.5.23 Uint Widget::GetHeight () const throw () `[virtual]`

Returns:

height

Get height of the Widget.

6.59.5.24 void Widget::SetWidth (Uint *_width*) throw () `[virtual]`

Parameters:

_width	new width

Set width of the Widget.

Note:

If entered size is bigger than *GetMaxSize()* or smaller than *GetMinSize()*, it will crop the entered value to the boundaries.

6.59.5.25 Uint Widget::GetWidth () const throw () `[virtual]`

Returns:

width

Get width of the Widget.

6.59.5.26 void Widget::SetGeometry (const Position & *_pos*, const Size & *_size*) throw (ParentNotDefined) `[virtual]`

Parameters:

_pos position new position

_size new size

Set both position and size of the Widget regarding to the *parentWindow*.

Note:

If entered size is bigger than *GetMaxSize()* or smaller than *GetMinSize()*, it will crop the entered value to the boundaries.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.27 void Widget::SetGeometry (Uint *_row*, Uint *_col*, Uint *_height*, Uint *_width*) throw (ParentNotDefined) `[virtual]`

Parameters:

_row new row position

_col new column position

_height new height

_width new width

Set both position and size of the Widget regarding to the *parentWindow*.

Note:

If entered size is bigger than *GetMaxSize()* or smaller than *GetMinSize()*, it will crop the entered value to the boundaries.

Exceptions:

ParentNotDefined is thrown had the widget not been assigned to any window. Use *AddWidget*.

6.59.5.28 void Widget::SetMinSize (const Size & *_size*) throw () `[virtual]`

Parameters:

_size new minimal size

Set minimal size of the Widget, *minSize* property.

Note:

If size is bigger than *GetMaxSize()*, it will crop the entered value to the boundary.

6.59.5.29 void Widget::SetMinSize (Uint *_height*, Uint *_width*) throw () `[virtual]`

Parameters:

_height new minimal height
_width new minimal width

Set minimal size of the Widget, *minSize* property.

Note:

If size is bigger than *GetMaxSize()*, it will crop the entered value to the boundary.

6.59.5.30 const Size & Widget::GetMinSize () const throw () `[virtual]`

Returns:

minimal size

Get minimal size of the Widget.

6.59.5.31 void Widget::SetMinHeight (Uint *_height*) throw () `[virtual]`

Parameters:

_height new minimal height

Set minimal height of the Widget, *minSize* property.

Note:

If size is bigger than *GetMaxSize()*, it will crop the entered value to the boundary.

6.59.5.32 Uint Widget::GetMinHeight () const throw () `[virtual]`

Returns:

minimal height

Get minimal height of the Widget.

6.59.5.33 void Widget::SetMinWidth (Uint *_width*) throw () `[virtual]`

Parameters:

_width new minimal width

Set minimal width of the Widget, *minSize* property.

Note:

If size is bigger than *GetMaxSize()*, it will crop the entered value to the boundary.

6.59.5.34 Uint Widget::GetMinWidth () const throw () `[virtual]`

Returns:

minimal width

Get minimal width of the Widget.

6.59.5.35 void Widget::SetMaxSize (const Size & *_size*) throw () `[virtual]`

Parameters:

_size new maximal size

Set maximal size of the Widget, *minSize* property.

Note:

If size is smaller than *GetMinSize()*, it will crop the entered value to the boundary.

6.59.5.36 void Widget::SetMaxSize (Uint *_height*, Uint *_width*) throw () `[virtual]`

Parameters:

_height new maximal height
_width new maximal width

Set maximal size of the Widget, *minSize* property.

Note:

If size is smaller than *GetMinSize()*, it will crop the entered value to the boundary.

6.59.5.37 const Size & Widget::GetMaxSize () const throw () `[virtual]`

Returns:

maximal size

Get maximal size of the Widget.

6.59.5.38 void Widget::SetMaxHeight (Uint *_height*) throw () `[virtual]`

Parameters:

_height new maximal height

Set maximal height of the Widget, *minSize* property.

Note:

If size is smaller than *GetMinSize()*, it will crop the entered value to the boundary.

6.59.5.39 Uint Widget::GetMaxHeight () const throw () `[virtual]`

Returns:

maximal height

Get maximal height of the Widget.

6.59.5.40 void Widget::SetMaxWidth (Uint *_width*) throw () `[virtual]`

Parameters:

_width new maximal width

Set maximal width of the Widget, *minSize* property.

Note:

If size is smaller than *GetMinSize()*, it will crop the entered value to the boundary.

6.59.5.41 Uint Widget::GetMaxWidth () const throw () `[virtual]`

Returns:

maximal width

Get maximal width of the Widget.

6.59.5.42 void Widget::SetFocusPolicy (FocusPolicy *_policy*) throw () `[virtual]`

Parameters:

_policy new focus policy

Set focus policy.

6.59.5.43 Widget::FocusPolicy Widget::GetFocusPolicy () const throw () `[virtual]`

Returns:

current focus policy

Get current focus policy.

6.59.5.44 void Widget::SetStyle (const DisplayStyle & *style* = DisplayStyle `(Fg::System,Fg::Dark, Bg::System)`) throw () `[virtual]`

Parameters:

style Set style.

6.59.5.45 const DisplayStyle & Widget::GetStyle () const throw () `[virtual]`

Returns:

current style

Get style.

6.59.5.46 void Widget::SetHidden (bool *_hidden*) throw ()

Parameters:

_hidden new state value

Set the hidden state.

6.59.5.47 bool Widget::IsHidden () const throw ()

Returns:

current hidden state

6.59.5.48 bool Scr::Tk::Widget::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented in Scr::Tk::ActiveWidget, Scr::Tk::BoxGroup, Scr::Tk::Checkbox, Scr::Tk::FramedWindow, Scr::Tk::HorizontalGroup, Scr::Tk::Inputbox, Scr::Tk::Label, Scr::Tk::RootWindow, Scr::Tk::ScrollbarBase, Scr::Tk::HorizontalScrollbar, Scr::Tk::VerticalScrollbar, Scr::Tk::Selectbox, Scr::Tk::VerticalGroup, Scr::Tk::VirtualWindow< W >, Scr::Tk::WidgetGroup, Scr::Tk::Window, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.59.5.49 const char $*$ Scr::Tk::Widget::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented in Scr::Tk::ActiveWidget, Scr::Tk::BoxGroup, Scr::Tk::Checkbox, Scr::Tk::FramedWindow, Scr::Tk::HorizontalGroup, Scr::Tk::Inputbox, Scr::Tk::Label, Scr::Tk::RootWindow, Scr::Tk::ScrollbarBase, Scr::Tk::HorizontalScrollbar, Scr::Tk::VerticalScrollbar, Scr::Tk::Selectbox, Scr::Tk::VerticalGroup, Scr::Tk::VirtualWindow< W >, Scr::Tk::WidgetGroup, Scr::Tk::Window, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.59.5.50 const char * Scr::Tk::Widget::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented in Scr::Tk::ActiveWidget, Scr::Tk::BoxGroup, Scr::Tk::Checkbox, Scr::Tk::FramedWindow, Scr::Tk::HorizontalGroup, Scr::Tk::Inputbox, Scr::Tk::Label, Scr::Tk::RootWindow, Scr::Tk::ScrollbarBase, Scr::Tk::HorizontalScrollbar, Scr::Tk::VerticalScrollbar, Scr::Tk::Selectbox, Scr::Tk::VerticalGroup, Scr::Tk::VirtualWindow< W >, Scr::Tk::WidgetGroup, Scr::Tk::Window, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.59.5.51 const Widget::ClassHierarchy & Scr::Tk::Widget::Hierarchy () `[inline]`

Returns:

class hierarchy of this widget.

6.59.6 Member Data Documentation

6.59.6.1 Window* Scr::Tk::Widget::parentWindow `[private]`

All widgets have a pointer to their parent window. For *RootWindow*, it is a pointer to itself

Note:

For not assigned widgets it is NULL.

6.59.6.2 Stylesheet* Scr::Tk::Widget::styleSheet `[private]`

Pointer to the stylesheet. If NULL, the widget's properties should be left default.

6.59.6.3 Position Scr::Tk::Widget::position `[protected]`

Position regarding the *parentWindow*. i.e. (position.row == 3) means that row 3 of *parentWindow* is 0th row of this widget.

6.59.6.4 Size Scr::Tk::Widget::size `[protected]`

Current size.

6.59.6.5 Size Scr::Tk::Widget::sizeMax `[protected]`

Maximal size that the Widget can be expanded to by for example a *WidgetGroup*.

6.59.6.6 Size Scr::Tk::Widget::sizeMin [protected]

Minimal size that the Widget can be shrinked to by for example a *WidgetGroup*.

6.59.6.7 DisplayStyle Scr::Tk::Widget::style [protected]

Basic style.

6.59.6.8 bool Scr::Tk::Widget::hidden [protected]

Implies whether the element is hidden. /note When hidden, the element want be a subject into positioning algorithms and its OnRedraw event won't invoked.

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

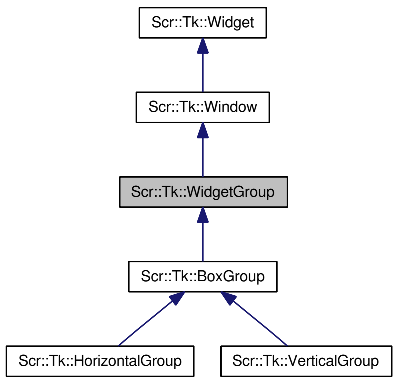
- include/rexio/tk/widget.h++
- lib/toolkit/src/widget.c++

6.60 Scr::Tk::WidgetGroup Class Reference

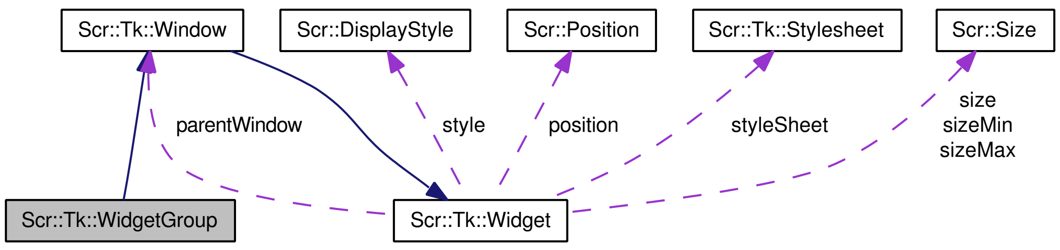
General class for grouping widgets and managing them.

`#include <widgetgroup.h++>`

Inheritance diagram for Scr::Tk::WidgetGroup:



Collaboration diagram for Scr::Tk::WidgetGroup:



Public Member Functions

- virtual void SwapWidgets (Widget &widget1, Widget &widget2) throw ()
- virtual void ShiftFWidget (Widget &widget) throw ()
- virtual void ShiftBWidget (Widget &widget) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Member Functions

- virtual void ArrangeContents () throw ()

6.60.1 Detailed Description

General class for grouping widgets and managing them.

This class is a base class for all sorts of of grouping widgets. Widgets inside of

6.60.2 Member Function Documentation

6.60.2.1 void WidgetGroup::ArrangeContents () throw () `[protected, virtual]`

where all magic is done :)

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

6.60.2.2 void WidgetGroup::SwapWidgets (Widget & *widget1*, Widget & *widget2*) throw () `[virtual]`

Parameters:

widget1 First widget
widget2 Second widget

Swap two widgets with together, provided that they are being contained by the WidgetGroup.

Reimplemented in Scr::Tk::BoxGroup.

6.60.2.3 void WidgetGroup::ShiftFWidget (Widget & *widget*) throw () `[virtual]`

Parameters:

widget Targetted widget

Move the widget further away on the containing widget list. Upon end of the list, move to the beginning.

6.60.2.4 void WidgetGroup::ShiftBWidget (Widget & *widget*) throw () `[virtual]`

Parameters:

widget Targetted widget

Move the widget closer on the containing widget list. Upon beginning of the list, move to the end.

6.60.2.5 virtual bool Scr::Tk::WidgetGroup::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className	name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::Window.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

6.60.2.6 virtual const char∗ Scr::Tk::WidgetGroup::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::Window.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

6.60.2.7 virtual const char∗ Scr::Tk::WidgetGroup::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::Window.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::HorizontalGroup, and Scr::Tk::VerticalGroup.

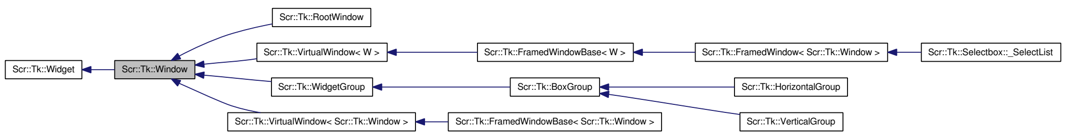
The documentation for this class was generated from the following files:

- include/rexio/tk/widgetgroup.h++
- lib/toolkit/src/widgetgroup.c++

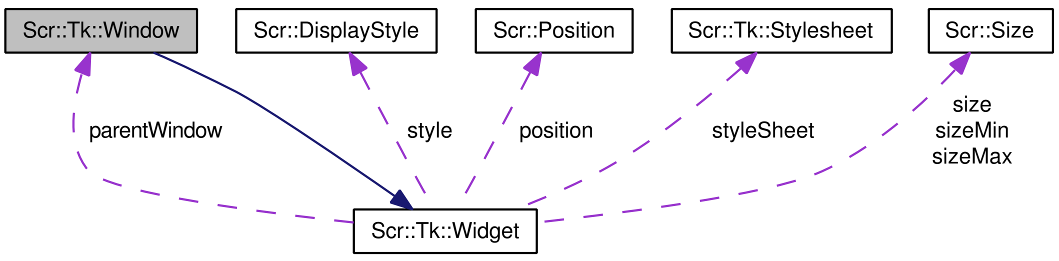
6.61 Scr::Tk::Window Class Reference

`#include <window.h++>`

Inheritance diagram for Scr::Tk::Window:


Scr::Tk::Widget
Scr::Tk::Window
Scr::Tk::RootWindow
Scr::Tk::VirtualWindow< W >
Scr::Tk::FramedWindowBase< W >
Scr::Tk::FramedWindow< Scr::Tk::Window >
Scr::Tk::Selectbox::_SelectList
Scr::Tk::WidgetGroup
Scr::Tk::BoxGroup
Scr::Tk::HorizontalGroup
Scr::Tk::VerticalGroup
Scr::Tk::VirtualWindow< Scr::Tk::Window >
Scr::Tk::FramedWindowBase< Scr::Tk::Window >

Collaboration diagram for Scr::Tk::Window:



Public Member Functions

- virtual Uint GetAbsoluteColumn () throw (ParentNotDefined)
- virtual Uint GetAbsoluteRow () throw (ParentNotDefined)
- Window (Uint _height, Uint _width, const DisplayStyle &_style=DisplayStyle(Fg::White, Fg::Dark, Bg::Black)) throw ()
- virtual void SetStylesheet (Stylesheet ∗_styleSheet) throw ()
- virtual void AddWidget (Widget &widget) throw (ParentAlreadySet, WidgetAlreadyAdded)
- virtual void DelWidget (Widget &widget) throw (WidgetNotPresent)
- virtual RootWindow & GetRootWindow () throw (ParentNotDefined)
- virtual void RedrawRequest () throw ()
- virtual void RedrawRequest (Widget &widget) throw ()
- virtual void OnFocus (FocusPolicy focustype) throw ()
- virtual void OnUnFocus (FocusPolicy focustype) throw ()
- virtual void PassFocusRequest (FocusPolicy focustype) throw ()
- virtual void SetActiveWidget (Widget &w) throw (WidgetNotPresent)
- virtual Widget & GetActiveWidget () const throw (WidgetNotPresent)
- virtual void OnResize () throw ()
- virtual void OnRedraw (Screen &screen) throw ()
- virtual void OnKeyDown (Key key) throw ()
- virtual void SetSize (const Size &_size) throw ()
- virtual bool IsTypeOf (std::string _className) const
- virtual const char ∗ TypeName () const
- virtual const char ∗ ParentName () const

Protected Types

- typedef AutoList< Widget ∗ > WidgetList

Protected Member Functions

- void NextWidget ()
- virtual Screen & GetScreen () throw (ParentNotDefined)

Protected Attributes

- WidgetList elements
- WidgetList::iterator activeWidget

6.61.1 Detailed Description

Window, a buffered ancestor of *Widget*. It can also group other widgets and pass all the events down the path.

See also:

WidgetGroup for an automated Widget grouping solution.

6.61.2 Member Typedef Documentation

6.61.2.1 typedef AutoList<Widget*> Scr::Tk::Window::WidgetList `[protected]`

Widget dedicated container.

6.61.3 Constructor & Destructor Documentation

6.61.3.1 Window::Window (Uint *_height*, Uint *_width*, const DisplayStyle & *_style* = DisplayStyle`(Fg::White,Fg::Dark,Bg::Black)`) throw ()

Parameters:

_height desired height

_width desired width

_style optional style

6.61.4 Member Function Documentation

6.61.4.1 void Scr::Tk::Window::NextWidget () `[protected]`

Focuses on a next contained element that has a proper *focusPolicy*. Specifically, *activeWidget* iterator is incremented.

6.61.4.2 Screen & Window::GetScreen () throw (ParentNotDefined) `[protected, virtual]`

Returns:

Screen handler reference.

Returns the top-level Screen handler.

Exceptions:

ParentNotDefined is thrown had the window not been attached to any other.

Reimplemented in Scr::Tk::RootWindow.

6.61.4.3 Uint Window::GetAbsoluteColumn () throw (ParentNotDefined) `[virtual]`

Returns an absolute column the window is positioned on a RootWindow

Exceptions:

> ***ParentNotDefined*** is thrown had the window not been attached to any other.

Reimplemented in Scr::Tk::RootWindow.

6.61.4.4 Uint Window::GetAbsoluteRow () throw (ParentNotDefined) `[virtual]`

Returns an absolute row the window is positioned on a RootWindow

Exceptions:

> ***ParentNotDefined*** is thrown had the window not been attached to any other.

Reimplemented in Scr::Tk::RootWindow.

6.61.4.5 void Window::SetStylesheet (Stylesheet ∗ *_styleSheet*) throw () `[virtual]`

Parameters:

> ***_styleSheet*** pointer to style data

Apply Stylesheet to this widget. Reinitialize any style properties if their alternatives are supplied. *Window* specific: Recursively passes this call to all its children.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::FramedWindowBase< W >, and Scr::Tk::FramedWindowBase< Scr::Tk::Window >.

6.61.4.6 void Window::AddWidget (Widget & *widget*) throw (ParentAlreadySet, WidgetAlreadyAdded) `[virtual]`

Parameters:

> ***widget*** widget to attach to this window

Attach a widget to this window. Specifically, add it to the *elements*.

Exceptions:

> ***ParentAlreadySet*** is thrown if the widget has already been attached to some other window.
>
> ***WidgetAlreadyAdded*** if the widget is already attached to THIS window.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::VirtualWindow< W >, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.61.4.7 void Window::DelWidget (Widget & *widget*) throw (WidgetNotPresent) `[virtual]`

Parameters:

> ***widget*** widget to detach from this window

Detach a widget from this window. Specifically, del it from the *elements*.

Exceptions:

WidgetNotPresent is thrown if the widget is not attached to this window.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::VirtualWindow< W >, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.61.4.8 RootWindow & Window::GetRootWindow () throw (ParentNotDefined) `[virtual]`

Returns:

RootWindow

Exceptions:

ParentNotDefined is thrown if the window hasn't been attached to any other and thus is not in relation with the root one.

Reimplemented in Scr::Tk::RootWindow.

6.61.4.9 void Window::RedrawRequest () throw () `[virtual]`

Need to redraw, pass the *OnRedraw()* event to all contained widgets.

Reimplemented from Scr::Tk::Widget.

6.61.4.10 void Window::RedrawRequest (Widget & *widget*) throw () `[virtual]`

Parameters:

widget reference to widget which needs redrawing

Redraw one specific widget. Pass the *OnRedraw()* event to it.

6.61.4.11 void Window::OnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element focused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Selectbox::_SelectList.

6.61.4.12 void Window::OnUnFocus (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype Type of the event, i.e. mouse click.

Element unfocused. Only matters if a proper *focusPolicy* is set.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Selectbox::_SelectList.

6.61.4.13 void Window::PassFocusRequest (FocusPolicy *focustype*) throw () `[virtual]`

Parameters:

focustype focus policy of this event

This event is triggered when containing event does want to revoke its focus.

6.61.4.14 void Window::SetActiveWidget (Widget & *w*) throw (WidgetNotPresent) `[virtual]`

Parameters:

w widget to activate

Activates a given widget. Widget has to be directly contained by this window.

Note:

Widget might directly revoke its activity.

Exceptions:

WidgetNotPresent is thrown if the widget is not attached to this window.

6.61.4.15 Widget & Window::GetActiveWidget () const throw (WidgetNotPresent) `[virtual]`

Returns:

reference to current active widget

Exceptions:

WidgetNotPresent is thrown if no widget is currently active.

6.61.4.16 void Window::OnResize () throw () `[virtual]`

Resize event. Do something i.e. adjust content to the new size.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::FramedWindowBase< W >, Scr::Tk::Selectbox::_-SelectList, Scr::Tk::VirtualWindow< W >, Scr::Tk::FramedWindowBase< Scr::Tk::Window >, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.61.4.17 void Window::OnRedraw (Screen & *screen*) throw () `[virtual]`

Parameters:

screen reference to the screen on which to draw

This is the main thing, the core of the Widget. Upon this event, the whole content should be redrawn.

Note:

the screen parameter is not a real screen, it is a cutdown to our size screen or even some other overloaded screen flavour.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::FramedWindowBase< W >, Scr::Tk::RootWindow, Scr::Tk::VirtualWindow< W >, Scr::Tk::FramedWindowBase< Scr::Tk::Window >, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.61.4.18 void Window::OnKeyDown (Key *key*) throw () `[virtual]`

Parameters:

key keycode

Keyboard button press event.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::Selectbox::_SelectList.

6.61.4.19 void Window::SetSize (const Size & *_size*) throw () `[virtual]`

Parameters:

_size new size

Set size of the Window. Invoke *OnResize()* event afterwards.

Note:

If entered size is bigger than *GetMaxSize()* or smaller than *GetMinSize()*, it will crop the entered value to the boundaries.
Since all the other size functions depend on this one, all of them get the *OnResize()* event for free.

Reimplemented from Scr::Tk::Widget.

6.61.4.20 virtual bool Scr::Tk::Window::IsTypeOf (std::string *_className*) const `[inline, virtual]`

Parameters:

_className name of a class

Returns:

whether the _className is in class hierarchy of this' class.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::FramedWindow, Scr::Tk::HorizontalGroup, Scr::Tk::RootWindow, Scr::Tk::VerticalGroup, Scr::Tk::VirtualWindow< W >, Scr::Tk::WidgetGroup, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.61.4.21 virtual const char∗ Scr::Tk::Window::TypeName () const `[inline, virtual]`

Returns:

class name of this widget.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::FramedWindow, Scr::Tk::HorizontalGroup, Scr::Tk::RootWindow, Scr::Tk::VerticalGroup, Scr::Tk::VirtualWindow< W >, Scr::Tk::WidgetGroup, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.61.4.22 virtual const char∗ Scr::Tk::Window::ParentName () const `[inline, virtual]`

Returns:

parent class of this widget.

Reimplemented from Scr::Tk::Widget.

Reimplemented in Scr::Tk::BoxGroup, Scr::Tk::FramedWindow, Scr::Tk::HorizontalGroup, Scr::Tk::RootWindow, Scr::Tk::VerticalGroup, Scr::Tk::VirtualWindow< W >, Scr::Tk::WidgetGroup, and Scr::Tk::VirtualWindow< Scr::Tk::Window >.

6.61.5 Member Data Documentation

6.61.5.1 WidgetList Scr::Tk::Window::elements `[protected]`

Represensts all contained widgets, including subwindows.

6.61.5.2 WidgetList::iterator Scr::Tk::Window::activeWidget `[protected]`

Currently active widget.

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

- include/rexio/tk/window.h++
- lib/toolkit/src/window.c++

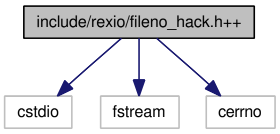
7 File Documentation

7.1 include/rexio/fileno_hack.h++ File Reference

extract file descriptor from C++ stream. Author of this code is Richard B. Kreckel

```
#include <cstdio>
#include <fstream>
#include <cerrno>
```

Include dependency graph for fileno_hack.h++:



Functions

- template<typename charT, typename traits>
  int fileno_hack (const std::basic_ios< charT, traits > &stream)

7.1.1 Detailed Description

extract file descriptor from C++ stream. Author of this code is Richard B. Kreckel

7.1.2 Function Documentation

7.1.2.1 template<typename charT, typename traits> int fileno_hack (const std::basic_ios< charT, traits > & *stream*) `[inline]`

Parameters:

stream a C++-style stream to extract FD from

Returns:

The integer file descriptor associated with the stream, or -1 if that stream is invalid. In the latter case, for the sake of keeping the code as similar to fileno(3), errno is set to EBADF.

See also:

The `upstream page at http://www.ginac.de/~kreckel/fileno/` of this code provides more detailed information.

Similar to fileno(3), but taking a C++ stream as argument instead of a FILE∗. Note that there is no way for the library to track what you do with the descriptor, so be careful.

7.2 include/rexio/throw.h++ File Reference

Useful macros for exception handling.

Defines

- #define __WHERE_AM_I__ "in " __FILE__ ":" TOSTRING(__LINE__)

  file name and line number as plain string

- #define THROW(x) throw x(__WHERE_AM_I__)

  throw exception x with __WHERE_AM_I__ as constructor argument

- #define EASSERT(assertion, exception) if (!(assertion))THROW(exception)

    throw exception when assertion evaluates false

- #define THROWP(x, p) throw x(std::string(__WHERE_AM_I__)+'\n'+(p))

    throw exception, that has specific parameters

- #define EASSERTP(a, e, p) if (!(a))THROWP(e,p)

    if assertion false, THROWP

7.2.1 Detailed Description

Useful macros for exception handling.

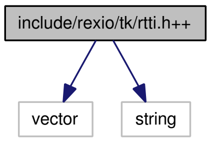
__WHERE_AM_I__ by Curtis Krauskopf; see whole article: `http://www.decompile.com/cpp/faq/file_-and_line_error_string.htm`

7.3 include/rexio/tk/rtti.h++ File Reference

`#include <vector>`

`#include <string>`

Include dependency graph for rtti.h++:



7.3.1 Detailed Description

RTTI - Run Time Type Information This macros can expand a class to have custom RTTI capabilities.

7.4 lib/screen/src/real/vt100codes.h++ File Reference

VT100 terminal control macros. Contains macro for cursor positioning, attribute setting, character sets etc. Used by Scr::VT100Compatible class.

Defines

- #define ENABLE_LINE_WRAP "\x1b[7h"

    enable line wrapping

- #define DISABLE_LINE_WRAP "\x1b[7l"

    disable it

- #define SCROLL_ENTIRE_SCREEN "\x1b[r"

Whole screen is scrolled on SCROLL_UP/SCROLL_DOWN.

- #define SCROLL_SCREEN_REGION(A, B) "\x1b["<< (A) << ';' << (B) << 'r'

  Only rows from A to B are scrolled on SCROLL_UP/SCROLL_DOWN, anything above A or below B is not scrolled.

- #define SCROLL_UP "\x1b[M"

  scroll up

- #define SCROLL_DOWN "\x1b[D"

  scroll down

- #define HIDE_CURSOR "\x1b[?25l"

  make cursor invisible - xterm

- #define SHOW_CURSOR "\x1b[?25h"

  restore it -xterm

- #define CURSOR_HOME "\x1b[H"

  Set cursor position to left-top position.

- #define CURSOR_YX(y, x) "\x1b["<< (y) << ';' << (x) << 'H'

  Set cursor position to specific y/x (note: y = 1..height, x = 1..width).

- #define CURSOR_UP "\x1b[A"

  move cursor one position up

- #define CURSOR_UP_(n) "\x1b["<< (n) <<'A'

  move cursor n positions up

- #define CURSOR_DOWN "\x1b[B"

  move cursor one position down

- #define CURSOR_DOWN_(n) "\x1b["<< (n) <<'B'

  move cursor n positions down

- #define CURSOR_FORWARD "\x1b[C"

  move cursor one position forward

- #define CURSOR_FORWARD_(n) "\x1b["<< (n) <<'C'

  move cursor n positions forward

- #define CURSOR_BACKWARD "\x1b[D"

  move cursor one position backward

- #define CURSOR_BACKWARD_(n) "\x1b["<< (n) <<'D'

  move cursor n positions backward

- #define SAVE_CURSOR "\x1b[s"

  One cursor position may be saved.

- #define UNSAVE_CURSOR "\x1b[u"

  and restored

- #define ERASE "\x1b[2J"

  Erase whole screen.

- #define ERASE_SCREEN ERASE

  same as above

- #define ERASE_UP "\x1b[1J"

  erase above cursor

- #define ERASE_DOWN "\x1b[J"

  erase below cursor

- #define ERASE_LINE "\x1b[K"

  erase current line

- #define ERASE_START_OF_LINE "\x1b[1K"

  erase current line left from the cursor

- #define ERASE_END_OF_LINE "\x1b[K"

  erase current line right from the cursor

- #define SET_ATTR(a) "\x1b["<<a<<'m'

  set specific attribute

- #define AND_ATTR <<';'<<

  if you have to set more attributes, separate them by <<';'<<

- #define ATTR_RESET 0

  resets terminal defaults

- #define ATTR_BRIGHT 1

  sets brighter fg color

- #define ATTR_DIM 2

  turns off bright (sets darker fg color) note: not supported by most of platforms

- #define ATTR_UNDERSCORE 4

  turns on text underline (not supported by MS Windows)

- #define ATTR_BLINK 5

  turns on blink (Not supported by MS Windows, most of other implementations incompatible)

- #define ATTR_REVERSE 7

  Inverts bg and fg color (incompatible implementation on MS windows)/.*

- #define CS_UK CS_UK_G0

Select UK character set.

- #define CS_US CS_US_G0

  Select US character set.

- #define CS_ALT CS_ALT_G0

  Select one of alt character set to use frames etc.

- #define RESIZE_SCREEN(A, B) "\x1b[8;"<< (A) << ";"<<(B) << "t"

  resize entire vscreen (xterm, konsole)

7.4.1 Detailed Description

VT100 terminal control macros. Contains macro for cursor positioning, attribute setting, character sets etc. Used by Scr::VT100Compatible class.

Index