

## CSIS2664 Documentation

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Packages . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Namespace Documentation</b>	<b>5</b>
3.1	Terminal Namespace Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.2	TerminalEditor Namespace Reference . . . . .	5
3.2.1	Detailed Description . . . . .	6
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	TerminalEditor.ITerminal Interface Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Member Function Documentation . . . . .	7
4.1.2.1	Clear() . . . . .	7
4.1.3	Property Documentation . . . . .	8
4.1.3.1	Height . . . . .	8
4.1.3.2	Input . . . . .	8
4.1.3.3	Navigator . . . . .	8
4.1.3.4	Output . . . . .	8
4.1.3.5	Width . . . . .	8
4.2	TerminalEditor.ITerminalFormatter Interface Reference . . . . .	8
4.2.1	Detailed Description . . . . .	8

4.2.2	Member Function Documentation	8
4.2.2.1	ResetColors()	8
4.2.2.2	SetBackground(ConsoleColor color)	8
4.2.2.3	SetForeground(ConsoleColor color)	9
4.3	TerminalEditor.ITerminalInput Interface Reference	9
4.3.1	Detailed Description	9
4.3.2	Member Function Documentation	9
4.3.2.1	Start()	9
4.3.2.2	Stop()	9
4.4	TerminalEditor.ITerminalNavigator Interface Reference	10
4.4.1	Detailed Description	10
4.4.2	Member Function Documentation	10
4.4.2.1	MoveCursorTo(int row, int col)	10
4.5	TerminalEditor.ITerminalWindow Interface Reference	10
4.5.1	Detailed Description	11
4.5.2	Member Function Documentation	11
4.5.2.1	ChangeArea(int x, int y, int width, int height)	11
4.5.2.2	Render()	11
4.5.3	Property Documentation	11
4.5.3.1	Height	11
4.5.3.2	Owner	11
4.5.3.3	Width	12
4.5.3.4	X	12
4.5.3.5	Y	12
4.6	TerminalEditor.ITextWriter Interface Reference	12
4.6.1	Detailed Description	12
4.6.2	Member Function Documentation	12
4.6.2.1	Write(string text, params string[] args)	12
4.6.2.2	WriteLine(string text, params string[] args)	13
4.6.3	Property Documentation	13

---

4.6.3.1	Formatter	13
4.7	Terminal.MyTerminal Class Reference	13
4.7.1	Detailed Description	14
4.7.2	Member Function Documentation	14
4.7.2.1	ClearTerminalWindow()	14
4.7.2.2	GetNextKey()	14
4.7.2.3	IsKeyAvailable()	14
4.7.2.4	MoveCursorTo(int col, int row)	14
4.7.2.5	ResetColors()	15
4.7.2.6	SetBackgroundColour(ConsoleColor color)	15
4.7.2.7	SetForegroundColour(ConsoleColor color)	15
4.7.2.8	WriteLineToTerminal(string text)	15
4.7.2.9	WriteToTerminal(string text)	15
4.7.3	Property Documentation	15
4.7.3.1	ColumnCount	15
4.7.3.2	RowCount	16
	<b>Index</b>	<b>17</b>



# Chapter 1

## Practical 2 Changelog

### **TerminalEditor** package

- Added the [TerminalEditor.IInputWindow](#) interface that should be implemented by terminal windows that can be used to input text.

### **TerminalComponents** package

- Added the [TerminalComponents](#) library with terminal components.





## Chapter 2

# Namespace Index

### 2.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">Terminal</a>	This namespace contains my implementation of a terminal emulator. . . . .	5
<a href="#">TerminalComponents</a>	NEW!! Library that provides a number of components that can be used to interact with the terminal window. . . . .	??
<a href="#">TerminalEditor</a>	( Updated )The core library of our terminal editor implementation. You will need to create implementations of all the interfaces provided by this library. . . . .	5



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

TerminalEditor.ITerminal . . . . .	7
TerminalEditor.ITerminalFormatter . . . . .	8
TerminalEditor.ITerminalInput . . . . .	9
TerminalEditor.ITerminalNavigator . . . . .	10
TerminalEditor.ITerminalWindow . . . . .	10
TerminalEditor.IInputWindow . . . . .	??
TerminalEditor.ITextWriter . . . . .	12
TerminalComponents.MultilineTextComponent . . . . .	??
Terminal.MyTerminal . . . . .	13
TerminalComponents.TextComponent . . . . .	??
TerminalComponents.TextInputComponent . . . . .	??



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">TerminalEditor.IInputWindow</a>	
( New ) Describes the structure of an interactive <a href="#">ITerminalWindow</a> instance. . . . .	??
<a href="#">TerminalEditor.ITerminal</a>	
This interface describes the structure of an object that will be used to interface with a terminal window or emulator. . . . .	7
<a href="#">TerminalEditor.ITerminalFormatter</a>	
This interface describes the structure of an object that will be used to format text written by an <a href="#">ITextWriter</a> object. . . . .	8
<a href="#">TerminalEditor.ITerminalInput</a>	
Describes the structure of an object that will be used by a <a href="#">ITerminal</a> object to read data from the keyboard. . . . .	9
<a href="#">TerminalEditor.ITerminalNavigator</a>	
Provides a means through which to move the cursor position in the terminal. . . . .	10
<a href="#">TerminalEditor.ITerminalWindow</a>	
Represents a window located within the terminal. This window can represent a text input area or so on. . . . .	10
<a href="#">TerminalEditor.ITextWriter</a>	
Describes the structure of an object that can be used by an <a href="#">ITerminal</a> instance to write text to a terminal window. . . . .	12
<a href="#">TerminalComponents.MultilineTextComponent</a>	
This class create a component that can be used to enter multiple lines of text . . . . .	??
<a href="#">Terminal.MyTerminal</a>	
My provided implementation of a terminal emulator. You need to incorporate this into your program. . . . .	13
<a href="#">TerminalComponents.TextComponent</a>	
Represents a component that can display text in a terminal window. This class cannot be inherited . . . . .	??
<a href="#">TerminalComponents.TextInputComponent</a>	
This class creates a single line component where text can be entered . . . . .	??



## Chapter 5

# Namespace Documentation

### 5.1 Terminal Namespace Reference

This namespace contains my implementation of a terminal emulator.

#### Classes

- class [MyTerminal](#)

*My provided implementation of a terminal emulator. You need to incorporate this into your program.*

#### 5.1.1 Detailed Description

This namespace contains my implementation of a terminal emulator.

### 5.2 TerminalComponents Namespace Reference

NEW!! Library that provides a number of components that can be used to interact with the terminal window.

#### Classes

- class [MultilineTextComponent](#)

*This class create a component that can be used to enter multiple lines of text*

- class [TextComponent](#)

*Represents a component that can display text in a terminal window. This class cannot be inherited*

- class [TextInputComponent](#)

*This class creates a single line component where text can be entered*

#### 5.2.1 Detailed Description

NEW!! Library that provides a number of components that can be used to interact with the terminal window.

## 5.3 TerminalEditor Namespace Reference

( Updated )The core library of our terminal editor implementation. You will need to create implementations of all the interfaces provided by this library.

### Classes

- interface [IInputWindow](#)  
*( New ) Describes the structure of an interactive [ITerminalWindow](#) instance.*
- interface [ITerminal](#)  
*This interface describes the structure of an object that will be used to interface with a terminal window or emulator.*
- interface [ITerminalFormatter](#)  
*This interface describes the structure of an object that will be used to format text written by an [ITextWriter](#) object.*
- interface [ITerminalInput](#)  
*Describes the structure of an object that will be used by a [ITerminal](#) object to read data from the keyboard.*
- interface [ITerminalNavigator](#)  
*Provides a means through which to move the cursor position in the terminal.*
- interface [ITerminalWindow](#)  
*Represents a window located within the terminal. This window can represent a text input area or so on.*
- interface [ITextWriter](#)  
*Describes the structure of an object that can be used by an [ITerminal](#) instance to write text to a terminal window.*

### 5.3.1 Detailed Description

( Updated )The core library of our terminal editor implementation. You will need to create implementations of all the interfaces provided by this library.



# Chapter 6

## Class Documentation

### 6.1 TerminalEditor.IInputWindow Interface Reference

( New ) Describes the structure of an interactive [ITerminalWindow](#) instance.

Inherits [TerminalEditor.ITerminalWindow](#).

Inherited by [TerminalComponents.InputWindowAdapter](#).

#### Public Member Functions

- void [WriteText](#) (string text)  
*Write text to the window at the current cursor position*
- void [ClearText](#) ()  
*Clear the window of all text*

#### Properties

- string [Text](#) [get]  
*Gets the text currently in the terminal window*
- int [CursorX](#) [get]  
*Gets current column of the cursor relative to the window's left.*
- int [CursorY](#) [get]  
*Gets current row of the cursor relative to the window's top*

#### 6.1.1 Detailed Description

( New ) Describes the structure of an interactive [ITerminalWindow](#) instance.

#### 6.1.2 Member Function Documentation

##### 6.1.2.1 void TerminalEditor.IInputWindow.ClearText ( )

Clear the window of all text

Implemented in [TerminalComponents.MultilineTextComponent](#), and [TerminalComponents.TextInputComponent](#).

##### 6.1.2.2 void TerminalEditor.IInputWindow.WriteText ( string text )

Write text to the window at the current cursor position

## Parameters

<i>text</i>	
-------------	--

Implemented in [TerminalComponents.MultilineTextComponent](#), and [TerminalComponents.TextInputComponent](#).

### 6.1.3 Property Documentation

6.1.3.1 `int TerminalEditor.IInputWindow.CursorX` [get]

Gets current column of the cursor relative to the window's left.

6.1.3.2 `int TerminalEditor.IInputWindow.CursorY` [get]

Gets current row of the cursor relative to the window's top

6.1.3.3 `string TerminalEditor.IInputWindow.Text` [get]

Gets the text currently in the terminal window

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

- `libraries/csharp/csieditor/IInputWindow.cs`

## 6.2 TerminalEditor.ITerminal Interface Reference

This interface describes the structure of an object that will be used to interface with a terminal window or emulator.

### Public Member Functions

- `void Clear ()`  
*Clear all text from the terminal*

### Properties

- `ITerminalInput Input` [get]  
*Gets the object to use to read input from the user*
- `ITextWriter Output` [get]  
*Gets the object to use to write to the terminal window*
- `ITerminalNavigator Navigator` [get]  
*Gets the object to use to move the terminal cursor around*
- `int Width` [get]  
*Number of columns in the terminal*
- `int Height` [get]  
*Number of rows in the terminal*

### 6.2.1 Detailed Description

This interface describes the structure of an object that will be used to interface with a terminal window or emulator.

### 6.2.2 Member Function Documentation

#### 6.2.2.1 void TerminalEditor.ITerminal.Clear ( )

Clear all text from the terminal

### 6.2.3 Property Documentation

#### 6.2.3.1 int TerminalEditor.ITerminal.Height [get]

Number of rows in the terminal

#### 6.2.3.2 ITerminalInput TerminalEditor.ITerminal.Input [get]

Gets the object to use to read input from the user

#### 6.2.3.3 ITerminalNavigator TerminalEditor.ITerminal.Navigator [get]

Gets the object to use to move the terminal cursor around

#### 6.2.3.4 ITextWriter TerminalEditor.ITerminal.Output [get]

Gets the object to use to write to the terminal window

#### 6.2.3.5 int TerminalEditor.ITerminal.Width [get]

Number of columns in the terminal

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

- libraries/csharp/csieditor/ITerminal.cs

## 6.3 TerminalEditor.ITerminalFormatter Interface Reference

This interface describes the structure of an object that will be used to format text written by an [ITextWriter](#) object.

## Public Member Functions

- void [SetForeground](#) (ConsoleColor color)
- void [SetBackground](#) (ConsoleColor color)  
*Set the background colour of the text buffer*
- void [ResetColors](#) ()  
*Reset the text buffer to use default colours*

### 6.3.1 Detailed Description

This interface describes the structure of an object that will be used to format text written by an [ITextWriter](#) object.

### 6.3.2 Member Function Documentation

#### 6.3.2.1 void TerminalEditor.ITerminalFormatter.ResetColors ( )

Reset the text buffer to use default colours

#### 6.3.2.2 void TerminalEditor.ITerminalFormatter.SetBackground ( ConsoleColor color )

Set the background colour of the text buffer

##### Parameters

<i>color</i>	
--------------	--

#### 6.3.2.3 void TerminalEditor.ITerminalFormatter.SetForeground ( ConsoleColor color )

Set the foreground colour of the text buffer

##### Parameters

<i>color</i>	
--------------	--

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

- libraries/csharp/csieditor/ITerminalFormatter.cs

## 6.4 TerminalEditor.ITerminalInput Interface Reference

Describes the structure of an object that will be used by a [ITerminal](#) object to read data from the keyboard.

## Public Member Functions

- void [Start](#) ()  
*Start listening for terminal input. This may be a blocking operation, so should only be used once all initialization is done.*
- void [Stop](#) ()  
*Stop waiting for terminal input. This should cause the application to exit*

### 6.4.1 Detailed Description

Describes the structure of an object that will be used by a [ITerminal](#) object to read data from the keyboard.

### 6.4.2 Member Function Documentation

#### 6.4.2.1 void TerminalEditor.ITerminalInput.Start ( )

Start listening for terminal input. This may be a blocking operation, so should only be used once all initialization is done.

#### 6.4.2.2 void TerminalEditor.ITerminalInput.Stop ( )

Stop waiting for terminal input. This should cause the application to exit

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

- libraries/csharp/csieditor/ITerminalInput.cs

## 6.5 TerminalEditor.ITerminalNavigator Interface Reference

Provides a means through which to move the cursor position in the terminal.

## Public Member Functions

- void [MoveCursorTo](#) (int row, int col)  
*Move the cursor to the specified row and column*

### 6.5.1 Detailed Description

Provides a means through which to move the cursor position in the terminal.

### 6.5.2 Member Function Documentation

#### 6.5.2.1 void TerminalEditor.ITerminalNavigator.MoveCursorTo ( int row, int col )

Move the cursor to the specified row and column

## Parameters

<i>row</i>	The row to move to	
<i>col</i>	The column to move to	

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

- `libraries/csharp/csieditor/ITerminalNavigator.cs`

## 6.6 TerminalEditor.ITerminalWindow Interface Reference

Represents a window located within the terminal. This window can represent a text input area or so on.

Inherited by `TerminalComponents.TerminalWindowComponent`, and [TerminalEditor.InputWindow](#).

### Public Member Functions

- void [ChangeArea](#) (int x, int y, int width, int height)  
*Change the location and size of the window*
- void [Render](#) ()  
*Draw the window in the terminal*

### Properties

- [ITerminal Owner](#) [get]  
*Gets the terminal object from which this window can get input from the user or write text to the terminal.*
- int [Width](#) [get]  
*Gets the width in columns of this window*
- int [Height](#) [get]  
*Gets the height in rows of this window*
- int [X](#) [get]  
*Gets the column coordinate of the top left corner*
- int [Y](#) [get]  
*Gets the row number of the top left corner*

#### 6.6.1 Detailed Description

Represents a window located within the terminal. This window can represent a text input area or so on.

#### 6.6.2 Member Function Documentation

##### 6.6.2.1 void TerminalEditor.ITerminalWindow.ChangeArea ( int x, int y, int width, int height )

Change the location and size of the window

## Parameters

<i>x</i>	The new column of the top left corner
<i>y</i>	The new row of the top left corner
<i>width</i>	The new width
<i>height</i>	The new height

## 6.6.2.2 void TerminalEditor.ITerminalWindow.Render ( )

Draw the window in the terminal

Implemented in [TerminalComponents.MultilineTextComponent](#), [TerminalComponents.TextComponent](#), and [TerminalComponents.TextInputComponent](#).

## 6.6.3 Property Documentation

## 6.6.3.1 int TerminalEditor.ITerminalWindow.Height [get]

Gets the height in rows of this window

## 6.6.3.2 ITerminal TerminalEditor.ITerminalWindow.Owner [get]

Gets the terminal object from which this window can get input from the user or write text to the terminal.

## 6.6.3.3 int TerminalEditor.ITerminalWindow.Width [get]

Gets the width in columns of this window

## 6.6.3.4 int TerminalEditor.ITerminalWindow.X [get]

Gets the column coordinate of the top left corner

## 6.6.3.5 int TerminalEditor.ITerminalWindow.Y [get]

Gets the row number of the top left corner

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

- `libraries/csharp/csieditor/ITerminalWindow.cs`

## 6.7 TerminalEditor.ITextWriter Interface Reference

Describes the structure of an object that can be used by an [ITerminal](#) instance to write text to a terminal window.

## Public Member Functions

- void [WriteLine](#) (string text, params string[] args)  
*Write a line of text followed by a new line to the terminal window*
- void [Write](#) (string text, params string[] args)  
*Write a line of text to the terminal window*

## Properties

- [ITerminalFormatter Formatter](#) [get]  
*Get the formatting object that can be used to customise the look of the text*

### 6.7.1 Detailed Description

Describes the structure of an object that can be used by an [ITerminal](#) instance to write text to a terminal window.

### 6.7.2 Member Function Documentation

#### 6.7.2.1 void TerminalEditor.ITextWriter.Write ( string text, params string[] args )

Write a line of text to the terminal window

##### Parameters

<i>text</i>	The text containing optional formatting
<i>args</i>	The arguments to add to the string formatting

Referenced by `TerminalComponents.MultilineTextComponent.Render()`.

#### 6.7.2.2 void TerminalEditor.ITextWriter.WriteLine ( string text, params string[] args )

Write a line of text followed by a new line to the terminal window

##### Parameters

<i>text</i>	The text containing optional formatting
<i>args</i>	The arguments to add to the string formatting

### 6.7.3 Property Documentation

#### 6.7.3.1 ITerminalFormatter TerminalEditor.ITextWriter.Formatter [get]

Get the formatting object that can be used to customise the look of the text

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



- `libraries/csharp/csieditor/ITextWriter.cs`

## 6.8 TerminalComponents.MultilineTextComponent Class Reference

This class create a component that can be used to enter multiple lines of text

Inherits `TerminalComponents.InputWindowAdapter`.

### Public Member Functions

- void `InsertLine` ()  
*Insert a new line at the current cursor position*
- void `Backspace` ()  
*Remove the character at the cursor position.*
- override void `Render` ()  
*Render the contents of the component to the terminal window*
- void `WriteLine` (string text)  
*Write a line of text to the component, and moves the cursor to the next line*
- override void `WriteText` (string text)  
*Create text to the component at the current cursor position*
- override void `ClearText` ()  
*Clear all text from the component*

### Properties

- override string `Text` [get]  
*Get the text currently stored in the component's buffer*
- string `this[int row]` [get, set]  
*Gets or sets the text at the specified row.*

#### 6.8.1 Detailed Description

This class create a component that can be used to enter multiple lines of text

#### 6.8.2 Member Function Documentation

##### 6.8.2.1 void TerminalComponents.MultilineTextComponent.Backspace ( )

Remove the character at the cursor position.

##### 6.8.2.2 override void TerminalComponents.MultilineTextComponent.ClearText ( )

Clear all text from the component

Implements `TerminalEditor.IInputWindow`.

#### 6.8.2.3 void TerminalComponents.MultilineTextComponent.InsertLine ( )

Insert a new line at the current cursor position

Referenced by TerminalComponents.MultilineTextComponent.WriteLine().

#### 6.8.2.4 override void TerminalComponents.MultilineTextComponent.Render ( )

Render the contents of the component to the terminal window

Implements [TerminalEditor.ITerminalWindow](#).

References TerminalEditor.ITextWriter.Write().

#### 6.8.2.5 void TerminalComponents.MultilineTextComponent.WriteLine ( string text )

Write a line of text to the component, and moves the cursor to the next line

Parameters

<i>text</i>	
-------------	--

References TerminalComponents.MultilineTextComponent.InsertLine().

#### 6.8.2.6 override void TerminalComponents.MultilineTextComponent.WriteText ( string text )

Create text to the component at the current cursor position

Parameters

<i>text</i>	
-------------	--

Implements [TerminalEditor.IInputWindow](#).

### 6.8.3 Property Documentation

#### 6.8.3.1 override string TerminalComponents.MultilineTextComponent.Text [get]

Get the text currently stored in the component's buffer

#### 6.8.3.2 string TerminalComponents.MultilineTextComponent.this[int row] [get], [set]

Gets or sets the text at the specified row.

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

- libraries/csharp/components/MultilineInputComponent.cs

## 6.9 Terminal.MyTerminal Class Reference

My provided implementation of a terminal emulator. You need to incorporate this into your program.

### Public Member Functions

- void [ClearTerminalWindow](#) ()  
*Removes all text from the terminal window*
- ConsoleKeyInfo [GetNextKey](#) ()  
*Reads the next available key from the key buffer*
- bool [IsKeyAvailable](#) ()  
*Checks if there is a key in the buffer*
- void [WriteLineToTerminal](#) (string text)  
*Write a line of text to the buffer and move the cursor to the next line.*
- void [WriteToTerminal](#) (string text)  
*Write text to the terminal.*
- void [ResetColors](#) ()  
*Reset the colour scheme to the default.*
- void [SetBackgroundColour](#) (ConsoleColor color)  
*Change the background colour of any subsequent text written to the terminal*
- void [SetForegroundColour](#) (ConsoleColor color)  
*Change the foreground colour of any subsequent text written to the terminal*
- void [MoveCursorTo](#) (int col, int row)  
*Move the cursor to the specified column and row. Both should be positive or zero*

### Properties

- int [ColumnCount](#) [get]  
*Get the number of columns in the terminal window*
- int [RowCount](#) [get]  
*Get the number of rows in the terminal window*

#### 6.9.1 Detailed Description

My provided implementation of a terminal emulator. You need to incorporate this into your program.

#### 6.9.2 Member Function Documentation

##### 6.9.2.1 void Terminal.MyTerminal.ClearTerminalWindow ( )

Removes all text from the terminal window

#### 6.9.2.2 ConsoleKeyInfo Terminal.MyTerminal.GetNextKey ( )

Reads the next available key from the key buffer

Returns

#### 6.9.2.3 bool Terminal.MyTerminal.IsKeyAvailable ( )

Checks if there is a key in the buffer

Returns

#### 6.9.2.4 void Terminal.MyTerminal.MoveCursorTo ( int *col*, int *row* )

Move the cursor to the specified column and row. Both should be positive or zero

Parameters

<i>col</i>	The column coordinate
<i>row</i>	The row coordinate

#### 6.9.2.5 void Terminal.MyTerminal.ResetColors ( )

Reset the colour scheme to the default.

#### 6.9.2.6 void Terminal.MyTerminal.SetBackgroundColour ( ConsoleColor *color* )

Change the background colour of any subsequent text written to the terminal

Parameters

<i>color</i>	
--------------	--

#### 6.9.2.7 void Terminal.MyTerminal.SetForegroundColour ( ConsoleColor *color* )

Change the foreground colour of any subsequent text written to the terminal

Parameters

<i>color</i>	
--------------	--

#### 6.9.2.8 void Terminal.MyTerminal.WriteLineToTerminal ( string *text* )

Write a line of text to the buffer and move the cursor to the next line.

##### Parameters

<i>text</i>	
-------------	--

#### 6.9.2.9 void Terminal.MyTerminal.WriteToTerminal ( string *text* )

Write text to the terminal.

##### Parameters

<i>text</i>	
-------------	--

### 6.9.3 Property Documentation

#### 6.9.3.1 int Terminal.MyTerminal.ColumnCount [get]

Get the number of columns in the terminal window

#### 6.9.3.2 int Terminal.MyTerminal.RowCount [get]

Get the number of rows in the terminal window

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

- libraries/csharp/terminal/Terminal.cs

## 6.10 TerminalComponents.TextComponent Class Reference

Represents a component that can display text in a terminal window. This class cannot be inherited

Inherits TerminalComponents.TerminalWindowComponent.

### Public Member Functions

- override void [Render](#) ()  
*Render the text to the terminal window*

### Properties

- string [Text](#) [get, set]  
*Get or set the text displayed in the component*

### 6.10.1 Detailed Description

Represents a component that can display text in a terminal window. This class cannot be inherited

### 6.10.2 Member Function Documentation

#### 6.10.2.1 `override void TerminalComponents.TextComponent.Render ( )`

Render the text to the terminal window

Implements [TerminalEditor.ITerminalWindow](#).

References `TerminalComponents.TextComponent.Text`.

### 6.10.3 Property Documentation

#### 6.10.3.1 `string TerminalComponents.TextComponent.Text` `[get]`, `[set]`

Get or set the text displayed in the component

Referenced by `TerminalComponents.TextComponent.Render()`.

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

- `libraries/csharp/components/TextComponent.cs`

## 6.11 TerminalComponents.TextInputComponent Class Reference

This class creates a single line component where text can be entered

Inherits `TerminalComponents.InputWindowAdapter`.

### Public Member Functions

- `override void Render ()`  
*Draw the window in the terminal*
- `override void WriteText (string text)`  
*Write text to the component*
- `override void ClearText ()`  
*Remove all text from the component*

### 6.11.1 Detailed Description

This class creates a single line component where text can be entered

## 6.11.2 Member Function Documentation

### 6.11.2.1 override void TerminalComponents.TextInputComponent.ClearText ( )

Remove all text from the component

Implements [TerminalEditor.IInputWindow](#).

### 6.11.2.2 override void TerminalComponents.TextInputComponent.Render ( )

Draw the window in the terminal

Implements [TerminalEditor.ITerminalWindow](#).

### 6.11.2.3 override void TerminalComponents.TextInputComponent.WriteText ( string text )

Write text to the component

Parameters

<i>text</i>	
-------------	--

Implements [TerminalEditor.IInputWindow](#).

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

- `libraries/csharp/components/TextInputComponent.cs`





# Index

- ChangeArea
  - TerminalEditor::ITerminalWindow, 11
- Clear
  - TerminalEditor::ITerminal, 7
- ClearTerminalWindow
  - Terminal::MyTerminal, 14
- ColumnCount
  - Terminal::MyTerminal, 15
- Formatter
  - TerminalEditor::ITextWriter, 13
- GetNextKey
  - Terminal::MyTerminal, 14
- Height
  - TerminalEditor::ITerminal, 8
  - TerminalEditor::ITerminalWindow, 11
- Input
  - TerminalEditor::ITerminal, 8
- IsKeyAvailable
  - Terminal::MyTerminal, 14
- MoveCursorTo
  - Terminal::MyTerminal, 14
  - TerminalEditor::ITerminalNavigator, 10
- Navigator
  - TerminalEditor::ITerminal, 8
- Output
  - TerminalEditor::ITerminal, 8
- Owner
  - TerminalEditor::ITerminalWindow, 11
- Render
  - TerminalEditor::ITerminalWindow, 11
- ResetColors
  - Terminal::MyTerminal, 14
  - TerminalEditor::ITerminalFormatter, 8
- RowCount
  - Terminal::MyTerminal, 15
- SetBackground
  - TerminalEditor::ITerminalFormatter, 8
- SetBackgroundColour
  - Terminal::MyTerminal, 15
- SetForeground
  - TerminalEditor::ITerminalFormatter, 9
- SetForegroundColour
  - Terminal::MyTerminal, 15
- Start
  - TerminalEditor::ITerminalInput, 9
- Stop
  - TerminalEditor::ITerminalInput, 9
- Terminal, 5
- Terminal.MyTerminal, 13
- Terminal::MyTerminal
  - ClearTerminalWindow, 14
  - ColumnCount, 15
  - GetNextKey, 14
  - IsKeyAvailable, 14
  - MoveCursorTo, 14
  - ResetColors, 14
  - RowCount, 15
  - SetBackgroundColour, 15
  - SetForegroundColour, 15
  - WriteLineToTerminal, 15
  - WriteToTerminal, 15
- TerminalEditor, 5
- TerminalEditor.ITerminal, 7
- TerminalEditor.ITerminalFormatter, 8
- TerminalEditor.ITerminalInput, 9
- TerminalEditor.ITerminalNavigator, 10
- TerminalEditor.ITerminalWindow, 10
- TerminalEditor.ITextWriter, 12
- TerminalEditor::ITerminal
  - Clear, 7
  - Height, 8
  - Input, 8
  - Navigator, 8
  - Output, 8
  - Width, 8
- TerminalEditor::ITerminalFormatter
  - ResetColors, 8
  - SetBackground, 8
  - SetForeground, 9
- TerminalEditor::ITerminalInput
  - Start, 9
  - Stop, 9
- TerminalEditor::ITerminalNavigator
  - MoveCursorTo, 10
- TerminalEditor::ITerminalWindow
  - ChangeArea, 11
  - Height, 11
  - Owner, 11
  - Render, 11
  - Width, 11
  - X, 12

Y, [12](#)

TerminalEditor::ITextWriter

Formatter, [13](#)

Write, [12](#)

WriteLine, [13](#)

Width

TerminalEditor::ITerminal, [8](#)

TerminalEditor::ITerminalWindow, [11](#)

Write

TerminalEditor::ITextWriter, [12](#)

WriteLine

TerminalEditor::ITextWriter, [13](#)

WriteLineToTerminal

Terminal::MyTerminal, [15](#)

WriteToTerminal

Terminal::MyTerminal, [15](#)

X

TerminalEditor::ITerminalWindow, [12](#)

Y

TerminalEditor::ITerminalWindow, [12](#)