

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Screen Object

See Also [Example](#) [Properties](#) [Methods](#) [Events](#)

Manipulates forms according to their placement on the screen and controls the mouse pointer outside your application's forms at [run time](#). The **Screen** object is accessed with the keyword **Screen**.

Syntax

Screen

Remarks

The **Screen** object is the entire Windows desktop. Using the **Screen** object, you can set the **MousePointer** property of the **Screen** object to the hourglass pointer while a modal form is displayed.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Section Object (Data Report Designer)

See Also [Example](#) [Properties](#) [Methods](#) [Events](#)

The **Section** object represents a section of the Data Report designer.

Syntax

Section

Remarks

You must use standard collection syntax to retrieve a reference to a Section object and any controls on the Data Report designer. The example below retrieves the name of the third control on the second section of the Section object:

```
Debug.Print DataReport1.Sections(2).Controls(3).Name
```

© 2017 Microsoft

Visual Basic Reference

Sections Collection Example

The example prints the name of each **Section** object in the Data Report designer as well as the names of each control in the section. To try the example, create a Data Report. Place a **CommandButton** control on a form, and paste the code into the Declarations section of the code module. Press F5 to run the project and click the button.

```
Private Sub Command1_Click()  
    Dim sect, ctl  
    For Each sect In DataReport1.Sections  
        Debug.Print "Section", sect.Name  
        For Each ctl In sect.Controls  
            Debug.Print , ctl.Name  
        Next ctl  
    Next sect  
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

SelBookmarks Collection

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

A **SelBookmarks** collection contains a bookmark for each row selected in a **DataGrid** control.

Syntax

SelBookmarks

Remarks

Use the **SelBookmarks** property of the **DataGrid** control to return the **SelBookmarks** collection. Bookmarks are added to the **SelBookmarks** collection in the order they're selected. You can reposition the **DataGrid** control's current record pointer by setting the **Bookmark** property to one of the selected bookmarks in the **SelBookmarks** collection.

Use the **Add** method to add bookmarks to the **SelBookmarks** collection. Once a bookmark is appended to the **SelBookmarks** collection, it appears selected in the **DataGrid** control.

To remove a bookmark from the **SelBookmarks** collection, use the **Remove** method. Once a bookmark is removed from the **SelBookmarks** collection, it no longer appears selected in the **DataGrid** control.

The **SelBookmarks** collection supports the **Add** and **Remove** methods as well as the **Count** property. Using these methods and properties, you can manipulate the list of selected items in the **DataGrid** control. For example, you can programmatically select additional items by using the **Add** method, or determine the total number of selected items using the **Count** property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

SelectedControls Collection

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

A [collection](#) that allows access to all of the currently selected controls on an object.

Syntax

SelectedControls(*index*)

The placeholder *index* represents an integer with a range from 0 to SelectedControls.Count - 1.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

SelectedVBControls Collection

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

Returns a collection of currently selected controls on a component.

Syntax

SelectedVBControls

Remarks

You can use this collection to access all currently selected controls on a form. The code can step through the collection of controls or request a specific control.

This collection has the same specifications as the **VBControls** collection, except this collection doesn't implement the **Add** method. The default method for the **SelectedVBControls** collection is the **Item** method and is indexed with integers.

This collection replaces the **SelectedControlTemplates** collection from Visual Basic version 4.0.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

SelectedVBControlsEvents Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

Represents a source of events supported by all currently selected controls.

Syntax

SelectedVBControlsEvents

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Series Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

An item from a **SeriesCollection** collection that represents a group of data points on a chart.

Syntax

Series

© 2017 Microsoft

Series Object Example

The following example changes the appearance of the lines for all series in a three-dimensional line chart.

```
Private Sub Command1_Click()  
    Dim serX As Series  
    ' Change the chart type to 3D line and smoothing  
    ' each line.  
    MSChart1.chartType = VtChChartType3dLine  
    MSChart1.ColumnCount = 4  
    For Each serX In MSChart1.Plot.SeriesCollection  
        serX.ShowGuideLine(VtChAxisIdY) = True  
        serX.GuideLinePen.Style = VtPenStyleDitted  
        serX.Pen.Style = 4  
    Next  
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Series Collection

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

A collection of chart series.

Syntax

Series (*index*)

The **Series** collection syntax has these parts:

Part	Description
<i>index</i>	Integer. Identifies the series of the chart. Series are identified in the order of data grid columns, beginning with 1.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

SeriesCollection Collection

See Also [Example](#) [Properties](#) [Methods](#) [Events](#)

Provides information about the series that make up a chart.

Syntax

SeriesCollection (*index*)

The **SeriesCollection** collection syntax has these parts:

Part	Description
<i>index</i>	Identifies a specific series in the series collection.

© 2017 Microsoft

Series, SeriesCollection Collections Example

The following example hides all the series in a chart.

```
Private Sub Command1_Click()  
    Dim serX As Series  
    ' Hides All Series.  
    For Each serX In MSChart1.Plot.SeriesCollection  
        serX.Position.Hidden = True  
    Next  
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

SeriesMarker Object

See Also [Example](#) Properties Methods Events

Describes a marker that identifies all data points within one series on a chart.

Syntax

SeriesMarker

© 2017 Microsoft

SeriesMarker Object Example

The following example sets marker parameters for all series in a chart.

```
Private Sub Command1_Click()  
    Dim serX As series  
    ' Show markers and unshow the lines for all series.  
    MSChart1.chartType = VtChChartType2dLine  
    For Each serX In MSChart1.Plot.SeriesCollection  
        serX.SeriesMarker.Show = True  
        serX.ShowLine = False  
    Next  
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

SeriesPosition Object


See Also [Example](#) [Properties](#) [Methods](#) [Events](#)

The location where a chart series is drawn in relation to other series. If all series have the same order (position), then they are stacked.

Syntax

SeriesPosition

© 2017 Microsoft



This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Shadow Object

See Also [Example](#) [Properties](#) [Methods](#) [Events](#)

Holds information about the appearance of a shadow on a chart element.

Syntax

Shadow

© 2017 Microsoft

Shadow Object Example

The following example sets a shadow on a chart backdrop title.

```
Private Sub Command1_Click()  
    ' Show shadow for title.  
    With MSChart1.Title  
        .Location.Visible = True  
        .Text = "Chart Title"  
        .Backdrop.Frame.Width = 1  
        .Backdrop.Frame.FrameColor.Set 255, 0, 0  
        .Backdrop.Frame.Style = VtFrameStyleSingleLine  
        .Backdrop.Shadow.Style = VtShadowStyleDrop  
        .Backdrop.Shadow.Offset.x = 10  
        .Backdrop.Shadow.Offset.y = 10  
    End With  
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Shape Control

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **Shape** control is a graphical control displayed as a rectangle, square, oval, circle, rounded rectangle, or rounded square.

Syntax

Shape

Remarks

Use **Shape** controls at design time instead of, or in addition to, invoking **Circle** and **Line** methods at [run time](#). You can draw a **Shape** control in a container, but it can't act as a container. The effect of setting the **BorderStyle** property depends on the setting of the **BorderWidth** property. If **BorderWidth** isn't 1 and **BorderStyle** isn't 0 or 6, **BorderStyle** is set to 1.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Shape Control (Data Report Designer)

See Also [Example](#) [Properties](#) [Methods](#) [Events](#)

The **Shape** control is a graphical control displayed as a rectangle, square, oval, circle, rounded rectangle, or rounded square.

Syntax

Shape

Remarks

The Data Report designer version of the **Shape** control is conceptually similar to the standard Visual Basic **Shape** control in providing a method of displaying certain shapes on any application. However, the Data Report designer version does not support the **BorderWidth**, **FillColor**, or **FillStyle** properties.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Slider Control

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

A **Slider** control is a window containing a slider and optional tick marks. You can move the slider by dragging it, clicking the mouse to either side of the slider, or using the keyboard.

Syntax

Slider

Remarks

Slider controls are useful when you want to select a discrete value or a set of consecutive values in a range. For example, you could use a **Slider** to set the size of a displayed image by moving the slider to a given tick mark rather than by typing a number. To select a range of values, set the **SelectRange** property to **True**, and program the control to select a range when the SHIFT key is down.

The **Slider** control can be oriented either horizontally or vertically.

Distribution Note To use the **Slider** control in your application, you must add the MSCOMCTL.OCX file to the project. When distributing your application, install the MSCOMCTL.OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an ActiveX control to a project, see the *Programmer's Guide*.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

Split Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

A **Split** object represents a [split](#) within a **DataGrid** control.

Remarks

The **DataGrid** supports Excel-like splits that divide the grid into vertical panes to provide users with different views of a database. Each split is represented by a **Split** object and contains a group of adjacent columns that scroll as a unit. When a **DataGrid** object is created, it contains one **Split** object by default.

You can use splits to present your data in multiple vertical panes. The data panes (or splits) can display data, scroll (vertically) together or independently of each other, and display the same or different columns. You can also use splits to fix one or more columns from scrolling. Unlike other grid products, the fixed columns do not have to be at the left edge of the grid, but can be at the right edge or anywhere in the middle. You can even have multiple groups of fixed columns within a grid.

You cannot stretch a column wider than its split. If you resize a split to be smaller than a column you can still grab the "phantom" splitter bar.

Each **Split** object maintains its own Columns collection. These independent splits and columns provide you with very powerful and flexible data presentation capabilities.

As mentioned above, a grid (a **DataGrid** object) initially contains a single split. If additional splits are created, you can determine or set the current split (i.e., the split that has received focus) using the grid's **Split** property as follows:

```
' Read the zero-based index of the current split  
Variable% = DataGrid1.Split
```

```
' Set focus to the split with an index equal to  
' Variable%  
DataGrid1.Split = Variable%
```

Each split in a grid is a different view of the same data source, and each split behaves just like an independent grid. If you create additional **Split** objects without customizing any of the split properties, all splits will be identical and each will behave very much like the original grid with one split.

Some of the properties of the **DataGrid** control are the same as the properties of a **Split** object and are considered common. Changes made to a **DataGrid** control common property also change the same property of the current **Split** object and vice versa. For example, consider a grid with two splits, and assume that the current split index is 1 (i.e., the grid's **Split** property is set to 1). If you want to determine the marquee style in use, the following statements are identical:

```
marquee% = DataGrid1.MarqueeStyle  
marquee% = DataGrid1.Splits(1).MarqueeStyle
```

If the current split index is set to 1, then the following code is equivalent for setting the **MarqueeStyle** property to **dbgSolidCellBorder**:

```
DataGrid1.MarqueeStyle = dbgSolidCellBorder  
DataGrid1.Splits(1).MarqueeStyle = dbgSolidCellBorder
```

Note Common properties are unique to **DataGrid** objects and their associated **Split** objects. No other object pairs possess similar relationships.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

Splits Collection

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The Splits collection contains all stored **Split** objects in a **DataGrid** control.

Syntax

Splits(*index*)

Splits.Item(*index*)

Remarks

You can create splits at design time using the grid's UI-active context menu. At run time, you can create and remove splits using the Splits collection's **Add** and **Remove** methods. Each method takes a zero-based split index. The following code demonstrates adding and removing splits at run time:

```
' Create a Split object with index 0  
DataGrid1.Splits.Add 0  
' Remove the Split object with index 1  
DataGrid1.Splits.Remove 1
```

You can determine the number of splits in a grid using the Splits collection's **Count** property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MSTab Control

Visual Studio 6.0

SSTab Control

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **SSTab** control provides a group of tabs, each of which acts as a container for other controls. Only one tab is active in the control at a time, displaying the controls it contains to the user while hiding the controls in the other tabs.

Syntax

SSTab

Remarks

An **SSTab** control is like the dividers in a notebook or the labels on a group of file folders. Using an **SSTab** control, you can define multiple pages for the same area of a window or dialog box in your application. Using the properties of this control, you can:

- Determine the number of tabs.
- Organize the tabs into more than one row.
- Set the text for each tab.
- Display a graphic on each tab.
- Determine the style of tabs used.
- Set the size of each tab.

To use this control, you must first decide how you want to organize the controls you will place into various tabs. Set the **Tabs** and **TabsPerRow** properties to create the tabs and organize them into rows. Then select each tab at design time by clicking the tab. For each tab, draw the controls you want displayed when the user selects that tab. Set the **Caption**, **Picture**, **TabHeight**, and **TabMaxWidth** properties as needed to customize the top part of the tab.

At run time, users can navigate between tabs by either pressing CTRL+TAB or by using accelerator keys defined in the caption of each tab.

You can also customize the entire **SSTab** control using the **Style**, **ShowFocusRect**, **TabOrientation**, and **WordWrap** properties.

Distribution Note The **SSTab** control is found in the TABCTL32.OCX file. To use the **SSTab** control in your application, you must add the control's .OCX file to the project. When distributing your application, install the appropriate .OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an additional control to a project, see the *Visual Basic Programmer's Guide*.

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

StatLine Object

See Also [Example](#) Properties Methods Events

Describes how statistic lines are displayed on a chart.

Syntax

StatLine

© 2017 Microsoft

StatLine Object Example

The following example sets the color and pen parameters for a chart statistics line.

```
Private Sub Command1_Click()  
    ' Show all statistic lines for series 2.  
    MSChart1.chartType = VtChChartType2dLine  
    With MSChart1.plot.SeriesCollection(2).StatLine  
        .VtColor.Set 128, 128, 255  
        .Flag = VtChStatsMinimum Or VtChStatsMaximum _  
        Or VtChStatsMean Or VtChStatsStddev Or _  
        VtChStatsRegression  
        .Style(vtChStatsMinimum) = VtPenStyleDotted  
        .width = 2  
    End With  
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

StatusBar Control

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

A **StatusBar** control provides a window, usually at the bottom of a parent form, through which an application can display various kinds of status data. The **StatusBar** can be divided up into a maximum of sixteen **Panel** objects that are contained in a **Panels** collection.

Syntax

StatusBar

Remarks

A **StatusBar** control consists of **Panel** objects, each of which can contain text and/or a picture. Properties to control the appearance of individual panels include **Width**, **Alignment** (of text and pictures), and **Bevel**. Additionally, you can use one of seven values of the **Style** property to automatically display common data such as date, time, and keyboard states.

At design time, you can create panels and customize their appearance by setting values in the Panel tab of the Properties Page of the **StatusBar** control. At run time, the **Panel** objects can be reconfigured to reflect different functions, depending on the state of the application. For detailed information about the properties, events, and methods of **Panel** objects, see the **Panel** Object and **Panels** Collection topics.

A **StatusBar** control typically displays information about an object being viewed on the form, the object's components, or contextual information that relates to that object's operation. The **StatusBar**, along with other controls such as the **ToolBar** control, gives you the tools to create an interface that is economical and yet rich in information.

Distribution Note The **StatusBar** control is part of a group of custom controls that are found in the MSCOMCTL.OCX file. To use the **StatusBar** control in your application, you must add the MSCOMCTL.OCX file to the project. When distributing your application, install the MSCOMCTL.OCX file in the user's Microsoft Windows SYSTEM folder. For more information on how to add a custom control to a project, see the *Programmer's Guide*.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

StdDataFormat Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **StdDataFormat** object allows formatting to be applied to data as it is read from and written to a database.

Syntax

StdDataFormat

Remarks

The **StdDataFormat** object is an integral part of data binding. It is created for any bound control, regardless of whether you create one in code.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

StdDataFormat Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **StdDataFormat** object allows formatting to be applied to data as it is read from and written to a database.

Syntax

StdDataFormat

Remarks

The **StdDataFormat** object is an integral part of data binding. It is created for any bound control, regardless of whether you create one in code.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

StdDataFormats Collection

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **StdDataFormats** collection contains one or more **StdDataFormat** objects.

Remarks

Complex bound controls can be bound to multiple **StdDataFormat** objects. The **DataGrid**, for example, has one binding per column. The **StdDataFormats** collection provides a top-level object through which you can access multiple format objects.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

StdDataValue Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

Sets or returns the value after formatting is performed by the **StdDataFormat** object, or the value written to the database after any work done in the Unformat event.

Syntax

StdDataValue

Remarks

The **StdDataValue** object is passed to the Format and Unformat events of the **StdDataFormat** object. Using the **StdDataValue** object in these events allows you to provide formatting not supported by the **StdDataFormat** object.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

StdFont Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **StdFont** object contains information needed to format text for display in the interface of an application or for printed output.

Syntax

StdFont

Remarks

You frequently identify a **StdFont** object using the **Font** property of an object that displays text (such as a **Form** object or the **Printer** object).

To create a **StdFont** object, use code like the following:

```
Dim X As New StdFont
```

If you put a **TextBox** control named Text1 on a form, you can dynamically change its **StdFont** object to another using the **Set** statement, as in the following example:

```
Dim X As New StdFont
X.Bold = True
X.Name = "Arial"
Set Text1.Font = X
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

StdPicture Object

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **StdPicture** object enables you to manipulate [bitmaps](#), icons, [metafiles](#) enhanced metafiles, GIF, and JPEG images assigned to objects having a **Picture** property.

Syntax

StdPicture

Remarks

You frequently identify a **StdPicture** object using the **Picture** property of an object that displays graphics (such as a **Form** object or a **PictureBox** control). If you have a **PictureBox** control named Picture1, you can set its **Picture** property to a **StdPicture** object using the **Set** statement, as in the following example:

```
Dim X As New StdPicture
Set X = LoadPicture("c:\windows\circles.bmp")
Set Picture1.Picture = X
```

You can use an array of **StdPicture** objects to keep a series of graphics in memory without needing a form that contains multiple **PictureBox** or **Image** controls.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: SysInfo Control

Visual Studio 6.0

SysInfo Control

[See Also](#) [Example](#) [Properties](#) [Methods](#) [Events](#)

The **SysInfo** control allows you to respond to certain system messages sent to all applications by the operating system. Your application can then adapt to changes in the operating system if necessary.

Syntax

SysInfo

Remarks

Using the **SysInfo** control, you can monitor information provided by the operating system as well as respond to system-generated events. The features of this control fall into one of these groups:

- Events tied to changes in the system (DisplayChanged, TimeChanged, and SettingChanged events, for example).
- Power status events and properties (PowerSuspend, PowerResume events and **ACStatus** and **BatteryStatus** properties, for example).
- Plug and Play events (DeviceArrival, DeviceRemoveComplete events, for example).
- Operating system properties (**OSVersion** and **WorkAreaHeight** properties, for example).

© 2017 Microsoft