(About) Applies To

SSDBCombo
SSDBCommand
SSDBDropDown
SSDBData
SSDBGrid
SSDBOptSet

About SelBookmarks

The SelBookmarks collection represents a set of selected bookmark objects. Bookmarks are added to this collection whenever a user selects a row in the grid. If Multiselect is True, then each row selected will be added to the collection in the order in which the selection occured. Order is never based on the displayed order. When a row is unselected, that row is then removed from the SelBookmarks collection. It is dangerous and not recommended to ever store the ordinal position of a row within this collection.

You can also add bookmarks to the SelBookmarks collection through code. The following example will add the first five rows to the collection:

```
Dim i as integer

SSDBGrid1.MoveFirst ' Position at the first row
for i = 0 to 4
        SSDBGrid1.SelBookmarks.Add SSDBGrid1.Bookmark
        SSDBGrid1.MoveNext
next i
```

It is also easy to access the rows in the SelBookmarks collection without moving the current row position. For example, if there was a column in the grid called Amount and you wanted to add up all the rows that were selected to get a total, you could use the following code:

```
Dim nTotal as long
Dim nTotalSelRows as integer
Dim i as integer
Dim bkmrk as Variant ' Bookmarks are always defined as variants

nTotalSelRows = SSDBGrid1.SelBookmarks.Count

' In the following, get the bookmark of the selected rows

for i = 0 to nTotalSelRows
   bkmrk = SSDBGrid1.SelBookmarks(i)
   nTotal = nTotal + SSDBGrid1.Columns("Amount").CellValue(bkmrk)
next i

Debug.Print "The total amount = " & Format( nTotal, "Currency")
```

About StyleSets

To understand StyleSet objects and the StyleSets collection, you should become familiar with the concept of *collections*.

A StyleSet is an object that contains a set of visual properties. In Data Widgets, the Data Grid, Data Combo, and Data DropDown all make use of StyleSet objects.

Creating StyleSets

It is possible to create style sets through the Grid Editor. It is also possible, and sometimes more appropriate, to create StyleSets through code.

In the case of the Data Grid, different StyleSets can be created and applied to specific columns, groups, and headers. Each of these Stylesets can in-turn be given characteristics which make one stand out from the other. For example, a "Loss" column of a grid containing financial data can have its **BackColor** property set to ‘red’, while a "Profit" column of a grid can have its **BackColor** property set to "green".

The following is an example of how a StyleSet may be set up:

1. The StyleSet is first added to the StyleSets Collection as follows:
   SSDBGrid1.StyleSets.Add "Houston"
2. Once added, the properties of a StyleSet may be set as follows:
   SSDBGrid1.StyleSets("Houston").BackColor = RGB(255,255,0)

Applying StyleSets

Once the StyleSet is created, it can be applied to an object that has a **StyleSet** Property . The following code applies the ‘Houston’ StyleSet to the **Column** object of a Data Grid control:

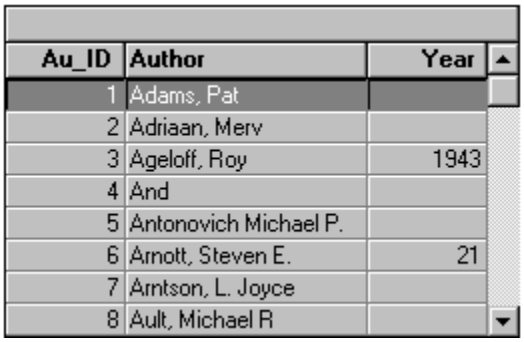
```
SSDBGrid1.Columns(1).StyleSet = "Houston"
```

Note If a change is made to a StyleSet, it does not have to be reapplied to an object to take effect. However, the control may need to be redrawn by invoking the **Refresh** method.

Achieving a 3D Look with the Data Combo

By setting just a few properties, you can quickly make your Data Combo have a 3D look to it. The following settings allow for a 3D look:

```
SSDBCombo1.BackColorEven = &H00C0C0C0&‘ Gray
SSDBCombo1.BackColorOdd = &H00C0C0C0&‘ Gray
SSDBCombo1.ForeColorEven = &H00000000& ‘ Black
SSDBCombo1.ForeColorOdd = &H00000000&  ‘ Black
SSDBCombo1.DividerStyle = 3 ‘ Inset
SSDBCombo1.DividerType = 3 ‘ Both
```



Au_ID	Author	Year
1	Adams, Pat	
2	Adriaan, Merv	
3	Ageloff, Roy	1943
4	And	
5	Antonovich Michael P.	
6	Arnott, Steven E.	21
7	Arntson, L. Joyce	
8	Ault, Michael R	

ActiveCell Applies To

SSDBGrid

ActiveCell Method See Also

ActiveCell Object

ActiveCell Object See Also

ActiveCell Method

ActiveRowStyleSet Property See Also

HeadStyleSet property
StyleSet property
StyleSet object
StyleSets collection

ActiveRowSyleSet Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Add Method Applies To

Bookmarks collection
Buttons collection
Columns collection
Groups collection
SelBookmarks collection
StyleSets collection

Add Method See Also

Bookmarks collection
Buttons collection
Columns collection
Groups collection
StyleSets collection
Count property
Remove method
RemoveAll method

AddItem Method (Column Object) Applies To

Column Object

AddItem Method (Column Object) See Also

Style
Columns collection

AddItem Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

AddItem Method See Also

DataMode
FieldDelimiter
FieldSeparator
RemoveItem method

AddItemBookmark Method

Applies To

Description

Returns the AddItem bookmark for a given absolute row number.

Syntax

object.**AddItemBookmark***(RowNum* **As Long***)*

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
RowNum	A numeric expression specifying the absolute row number.

Remarks

In bound mode, you are able to access bookmarks through the data control. This method gives the programmer access to the bookmarks for AddItem mode.

AddItemBookmark Method Applies To

SSDBGrid

Adding a Bound Data Combo

The Data Combo relies on the host environment's standard data control to access database information.

To use the Data Control with your application in Visual Basic:

1. Place two standard data controls on your form.
   One is used for the edit portion, the other is used for the list portion.
2. For both data controls, set the **DatabaseName** and **RecordSource** properties to point to a database and the table within the database.
3. Place a Data Combo control on your form.
4. Set the **DataSource** property of the Data Combo to point to the data control used for the edit portion. Set the **DataField** property to point to the field used.
5. Set the **DataSourceList** property of the Data Combo to point to the data control used for the list portion. Set the **DataFieldList** property to point to the field used.

Adding a Bound Data DropDown

Much like a Data Combo, the field in the cell is related to the list in the Data DropDown. To use the Data DropDown in a Data Grid:

1. Place two standard data controls on your form.
   One is used for the data grid, the other is used for the data combo.
2. For both data controls, set the **DatabaseName** and **RecordSource** properties to point to a database and the table within the database.
3. Place a Data Grid control on the form and bind it to the first data control.
4. Place a Data DropDown control on the form.
   The location of the Data DropDown is unimportant since it is invisible at runtime.
5. Set the **DataSource** property of the Data DropDown to the second data control.
6. Set the **DataFieldList** property of the Data DropDown to the field you want used for the list.
7. Link the Data DropDown to the Data Grid by adding the following code in the **InitColumnProps** procedure of the Data Grid:
   SSDBGrid1.Columns(n).DropDownhWnd = SSDBDropDown1.hWnd

Note The instructions above assume that the Data Grid is also bound to a data control. It is possible to have a bound Data DropDown work in conjunction with an unbound Data Grid. If the Data Grid is unbound, you will only need one data control, the one used for the Data DropDown.

Adding a Bound Data Grid to your application

The Data Grid makes use of the host environment's standard data control. To create a functional grid for your application in Visual Basic:

1. Add a Visual Basic Data Control to your form.
2. Set the **DatabaseName** and **RecordSource** properties in the data control.
3. Add a SSDBGrid Control to your form.
4. Set the **DataSource** property in the SSDBGrid control to the data control (i.e., Data1).

Your grid is now aware of the database associated with the data control. At this point, you can use the Grid Editor to design a grid format.

Adding a Data Command Button

The Data Command button only works when bound to a data control.

To use the Data Command button with your application in Visual Basic:

1. Place a standard data control on your form.
2. Set the **DatabaseName** and **RecordSource** properties to point to a database and the table within the database.
3. Place the Data Command button on your form.
4. Set the **DataSource** property of the Data Command button to point to the data control you created in Step 1.
5. Set the **DatabaseAction** property of the Data Command button to perform the action you want.

Adding an AddItem Grid to Your Application

In AddItem mode, you can add as many rows of data as you want, at any time during operation. This data is accessible as if the grid was bound (i.e., when you scroll, the next row of data is displayed automatically). Instead of a data control managing the flow of data, the grid does.

This mode operates similarly to the Visual Basic list box, but has all the features and power of the SSDBGrid. You can use the Grid Editor to help create the AddItem grid, or you can manually specify the properties.

Wherever possible, the grid in AddItem mode has the same functionality as a grid in bound mode, and most programmatic statements are the same.

The uses of this mode are virtually endless. One of its most useful features is being able to fill the grid with information without the need for a database. The AddItem mode is much better on system resources because it does not require the overhead of a Data Control. AddItem mode is best used for small lists that are easily maintained.

To create a an AddItem grid for your application:

1. Add a SSDBGrid control to your form.
2. Set the **DataMode** property to 2 (AddItem mode).
3. Specify the number of columns to use by setting the **Cols** property.
4. If you want to change the **FieldDelimiter** or **FieldSeperator** properties from their defaults, specify them now.
5. Specify code in the **InitColumnProps** event of the grid so that items are added when the grid first appears.

The following example demonstrates how **InitColumnProps** can be used to fill an AddItem grid:

```
Sub SSDBGrid1_InitColumnProps
Dim I As Integer
For I = 0 to 32
SSDBGrid1.AddItem "Hello" + CHR$(9) + "World"
Next I
End Sub
```

Adding an Unbound Data Combo

the Data Combo has two portions that it retrieves data for, with the ability to bind each part to different data sources. You can also configure the Data Combo to have either or both portions unbound, in which case, you will need to supply the data yourself.

When the edit portion of the Data Combo is unbound, you need to initially supply the field value yourself via the **Text** property which contains the value of the data in the edit portion of the Data Combo. When the user clicks on the dropdown button, the list portion will automatically update the **Text** property and the contents of the edit portion if the user selects a value, much like a standard combo box. The functionality of the Data Combo in unbound mode is identical to the Data Grid in unbound mode.

You can also set the Data Combo to AddItem mode, following the same guidelines used for the Data Grid in AddItem mode.

Adding an Unbound Data DropDown

The functionality of the Data DropDown in unbound mode is identical to the Data Grid in unbound mode.

You can also set the Data DropDown to AddItem mode, following the same guidelines used for the Data Grid when in this mode.

Adding an Unbound Data Grid to your application

The primary use of the Data Grid is to manage the display and entry of data into a record set of the bound data control. Because a database may contain an unlimited amount of data, the Data Grid has to manage the data in a virtual fashion, meaning that it only reads in as much data as it needs to display information on the screen.

Another important feature of the Data Grid is its ability to perform as an unbound control. Unbound mode is most useful when you need to handle data that the host environment's standard data control cannot. The only difference between bound and unbound mode is how the data is handled coming into the grid and going out of the grid.

The unbound grid sends cues in the form of events notifying you when it needs a response. When it needs more data, it fires the **UnboundReadData** event, likewise, when it needs to save data, it fires the **UnboundWriteData** event. Your primary responsibility in unbound mode is to supply the grid with data when it requests it, and to store data when it sends it.

To create an unbound grid for your application in Visual Basic:

1. Add a SSDBGrid Control to your form.
2. Set the **DataMode** to '1 - Unbound'.
3. Place code in the **UnboundReadData** event of the Data Grid that extracts data from your data source.
4. Place code in the **UnboundWriteData** event of the Data Grid that writes modified data back to your data source.
5. Place code in the **UnboundAddData** event of the Data Grid that appends a new row to your data source.
6. Place code in the **UnboundDeleteRow** event of the Data Grid that deletes a row from your data source.

Your unbound data grid is now ready for use.

Adding the DataOptionSet

Option buttons for the DataOptionSet can be created at either design or runtime. Once you have placed the control on the form, all you need to do is set properties that define the values for your DataOptionSet.

To use the DataOptionSet with your application in Visual Basic:

1. Place a standard data control on your form.
2. Set the **DatabaseName** and **RecordSource** properties to point to a database and the table within the database.
3. Place the DataOptionSet on your form.
4. Set the **DataSource** property of the DataOptionSet to point to the data control you created in Step 1.
5. Set the **DataField** property of the DataOptionSet so that it points to the field to work with.

The DataOptionSet has been added to your form, but you must create buttons at design time, or create buttons at run time in order for the control to be useful.

Adding the Enhanced Data Control

Adding the Enhanced Data Control to your form is quite simple. Remember that the EDC works in *conjunction* with the standard data control, not without it.

To use the Enhanced Data Control with your application:

1. Place a standard data control on your form.
2. Set the **DatabaseName** and **RecordSource** properties to point to a database and the table within the database.
3. Place the Enhanced Data Control on your form.
4. Set the **DataSource** property of the EDC to point to the data control you created in Step 1.
5. Set the **DataField** property of the EDC to point to the database field you want the EDC bound to.

AfterClick Event Applies To

SSDBCommand

AfterColUpdate Event Applies To

SSDBGrid

AfterColUpdate Event See Also

AfterInsert event
AfterUpdate event
BeforeColUpdate event
BeforeDelete event
BeforeInsert event
BeforeUpdate event

AfterDelete Event Applies To

SSDBGrid

AfterDelete Event See Also

AfterColUpdate event
AfterInsert event
AfterUpdate event
BeforeColUpdate event
BeforeDelete event
BeforeInsert event
BeforeUpdate event

AfterInsert Event Applies To

SSDBGrid

AfterInsert Event See Also

AfterColUpdate event
AfterDelete event
AfterUpdate event
BeforeColUpdate event
BeforeDelete event
BeforeInsert event
BeforeUpdate event

AfterUpdate Event Applies To

SSDBGrid

AfterUpdate Event See Also

AfterColUpdate event
AfterDelete event
AfterInsert event
BeforeColUpdate event
BeforeDelete event
BeforeInsert event
BeforeUpdate event

Alignment Property (Column Object) Applies To

Column Object

Alignment Property Applies To

SSDBData
SSDBOptSet

Alignment Property See Also

CaptionAlignment property

AllowAddNew Property Applies To

SSDBGrid

AllowAddNew Property See Also

AllowDelete
AllowUpdate

AllowColumnMoving Property Applies To

SSDBGrid

AllowColumnMoving Property See Also

AllowColumnSizing
AllowColumnSwapping

AllowColumnShrinking Property Applies To

SSDBGrid

AllowColumnShrinking Property See Also

AllowGroupShrinking

AllowColumnSizing Property Applies To

SSDBGrid

AllowColumnSizing Property See Also

AllowColumnMoving
AllowColumnSwapping

AllowColumnSwapping Property Applies To

SSDBGrid

AllowColumnSwapping Property See Also

AllowColumnMoving
AllowColumnSizing

AllowDelete Property Applies To

SSDBGrid

AllowDelete Property See Also

AllowAddNew
AllowUpdate

AllowDragDrop Property Applies To

SSDBGrid

AllowGroupMoving Property Applies To

SSDBGrid

AllowGroupMoving Property See Also

AllowGroupSizing
AllowGroupSwapping

AllowGroupShrinking Property Applies To

SSDBGrid

AllowGroupShrinking Property See Also

AllowColumnShrinking

AllowGroupSizing Property Applies To

SSDBGrid

AllowGroupSizing Property See Also

AllowGroupMoving
AllowGroupSwapping

AllowGroupSwapping Property Applies To

SSDBGrid

AllowGroupSwapping Property See Also

AllowGroupMoving
AllowGroupSizing

AllowInput Property Applies To

SSDBCombo

AllowInput Property See Also

AllowNull

AllowNull Property Applies To

SSDBCombo

AllowRowSizing Property Applies To

SSDBGrid

AllowSizing Property Applies To

Column object
Group object

AllowSizing Property See Also

AllowColumnSizing

AllowUpdate Property Applies To

SSDBGrid

AllowUpdate Property See Also

AllowAddNew
AllowDelete

Anatomy of a Data Combo

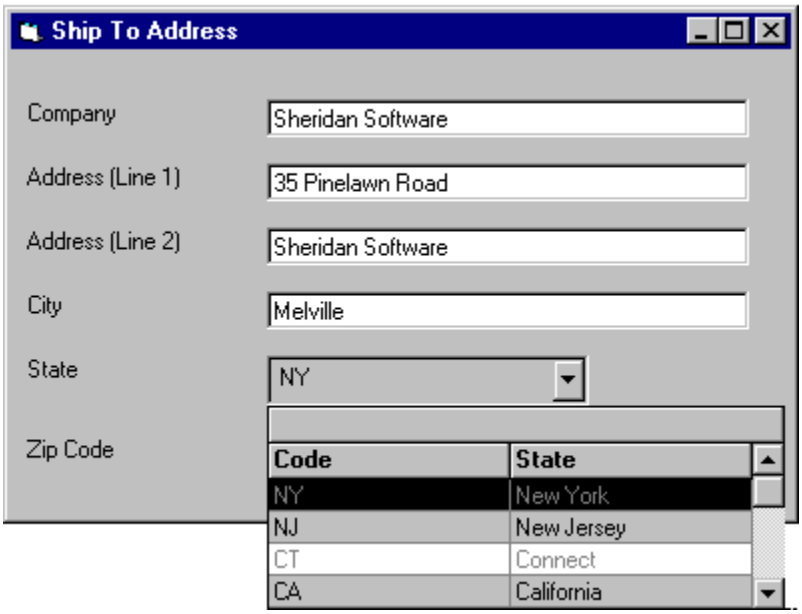
Although there are major fundamental differences, the Data Combo control looks and behaves much like a standard Windows combo box. The major difference is that the Data Combo can be bound to a data control.

The Data Combo is made up of two portions:

- The Edit portion of the Data Combo displays the selected field and allows entry.
- The List portion drops down when the user clicks the dropdown button.

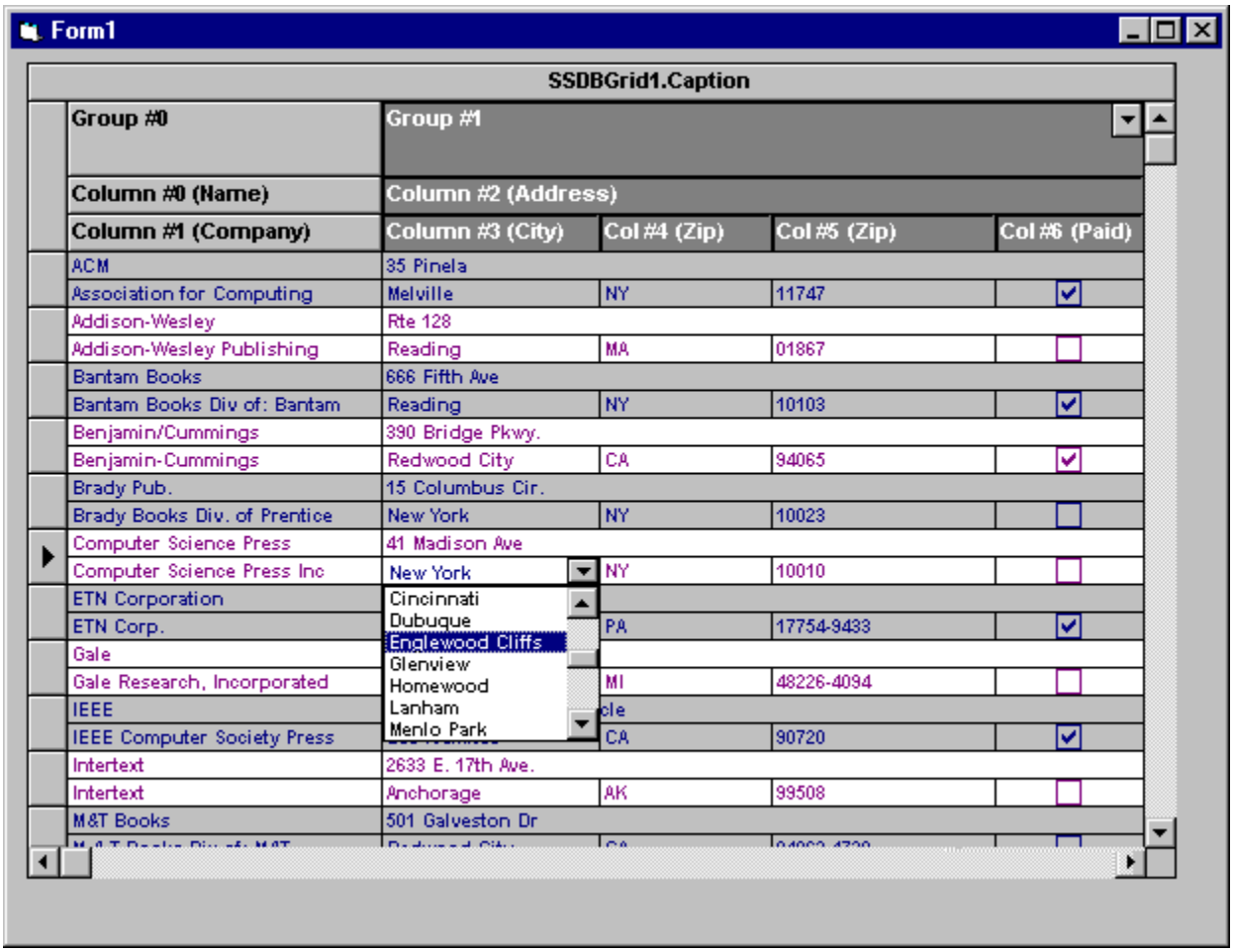
Typically, a combo box is used to allow entry of a particular field while allowing the user to select a value for that field via the dropdown list. With the Data Combo, you can bind the edit portion to a field in one database while the list portion can dropdown a list of values from another.

A classic example is to link the edit portion of the Data Combo to a field in a record set of a data control such as StateCode. Then, link the list portion of the Data Combo to a data control that manages a table having all state codes and translations. The results would look similar to:



Anatomy of a Data Grid

Click on an area of the Data Grid to learn more about it:



Anatomy of the Enhanced Data Control

First Record

Jumps to the first record in the database. This button is displayed/hidden by the **ShowFirstLastButtons** property.

Previous Page

Jumps to the previous page in the database. A page is determined by the setting of **PageValue**. This button is displayed/hidden by the **ShowPageButtons** property.

Previous Record

Jumps to the previous record in the database. This button is displayed/hidden by the **ShowPrevNextButtons** property.

Add Record

Adds a new record to the end of the database. This button is displayed/hidden by the **ShowAddButton** property.

Cancel Add

Cancels theadding of a new record to the database. This button is displayed/hidden by the **ShowCancelButton** property.

Delete Record

Deletes a record from the database. This button is displayed/hidden by the **ShowDeleteButton** property.

Update Record

Updates the selected record in the database. This button is displayed/hidden by the **ShowUpdateButton** property.

Add Bookmark

Adds a bookmark for the current record. This button is displayed/hidden by the **ShowBookmarksButton** property.

Clear All Bookmarks

Clears all stored bookmarks. This button is displayed/hidden by the **ShowBookmarksButton** property.

Current Record

When the **DataField** property is set, the active record is displayed. When **DataField** is left blank, the **Caption** is displayed.

Goto Bookmark

Presents a list of all stored bookmarks (up to a user-definable limit of 100). This button is displayed/hidden by the **ShowBookmarksButton** property.

Find Record

Invokes the Find dialog, allowing the user to search the database.

Find Previous Record

Searches backwards in the database for the next occurrence of data specified in the Find dialog.

Find Next Record

Searches forwards in the database for the next occurrence of data specified in the Find dialog.

Next Record

Jumps to the next record in the database. This button is displayed/hidden by the **ShowPrevNextButtons** property.

Next Page

Jumps to the next page in the database. A page is determined by the setting of **PageValue**. This button is displayed/hidden by the **ShowPageButtons** property.

Last Record

Jumps to the last record in the database. This button is displayed/hidden by the **ShowFirstLastButtons** property.

AutoRestore Property Applies To

SSDBCombo

AutoSize Property Applies To

SSDBCommand

BIBLIO File Structure

Visual Basic ships with a sample database file called BIBLIO.MDB, which is in Access 2.0 format. Due to the implementation of OCX compatibility by many development tools, it is very possible that users of Data Widgets may not be using Visual Basic as their host development environment.

In light of this fact, the following table describes the structure of the BIBLIO database so that those users can create a database for use with the examples given in this manual.

The BIBLIO database is made up of the following tables:

- Authors
- Publishers
- Title Author
- Titles

Authors

Au_ID	Unique key identifier
Author	Author's name
Year born	Author's birthdate

Publishers

PubID	Unique key identifier
Name	Short name
Company Name	Full business name
Address	Publisher's address
City	Publisher's city
State	Publisher's state
Zip	Publisher's zip code
Telephone	Publisher's phone number
Fax	Publisher's fax number
Comments	General comments

Title Author

ISBN	Foreign key into Titles table
Au_ID	Foreign key into Authors table

Titles

Title	Book title
Year Publisher	Publication date
ISBN	Unique key

PubID	Foreign key into publishers table
Description	Reference info
Notes	General notes
Subject	Keywords
Comments	Description of book contents

BackColor Property Applies To

Column object
SSDBCombo
SSDBDropDown
SSDBData
SSDBGrid
SSDBOptSet

BackColorEven Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

BackColorEven Property See Also

BackColor
BackColorOdd

BackColorOdd Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

BackColorOdd Property See Also

BackColor
BackColorEven

BalloonHelp Property Applies To

SSDBData
SSDBGrid

BeforeColUpdate Event Applies To

SSDBGrid

BeforeColUpdate Event See Also

AfterColUpdate
AfterDelete
AfterInsert
AfterUpdate
BeforeDelete
BeforeInsert
BeforeUpdate

BeforeDelete Event Applies To

SSDBGrid

BeforeDelete Event See Also

AfterColUpdate
AfterDelete
AfterInsert
AfterUpdate
BeforeColUpdate
BeforeInsert
BeforeUpdate

BeforeInsert Event Applies To

SSDBGrid

BeforeInsert Event See Also

AfterColUpdate
AfterDelete
AfterInsert
AfterUpdate
BeforeColUpdate
BeforeDelete
BeforeUpdate

BeforeUpdate Event Applies To

SSDBGrid

BeforeUpdate Event See Also

AfterColUpdate
AfterDelete
AfterInsert
AfterUpdate
BeforeColUpdate
BeforeDelete
BeforeInsert

BevelColorFace, BevelColorFrame, BevelColorHighlight, BevelColorShadow Properties Applies To

SSDBCombo
SSDBCommand
SSDBData
SSDBDropDown
SSDBGrid
SSDBOptSet

BevelColorScheme Property Applies To

SSDBCombo
SSDBCommand
SSDBData
SSDBDropDown
SSDBGrid
SSDBOptSet

BevelColorScheme Property See Also

BevelColorFace
BevelColorFrame
BevelColorHighlight
BevelColorShadow

BevelInner Property Applies To

SSDBData

BevelInner Property See Also

BevelColorFace
BevelColorFrame
BevelColorHighlight
BevelColorScheme
BevelColorShadow

BevelOuter Property Applies To

SSDBData

BevelOuter Property See Also

BevelInner
BevelColorHighlight
BevelColorScheme
BevelColorShadow

BevelType Property Applies To

SSDBCombo
SSDBDropDown

BevelWidth Property Applies To

SSDBCombo
SSDBCommand
SSDBData

BevelWidth Property See Also

BevelInner
BevelOuter

Binding a Data Command to a Data Control Across Forms

Starting with Visual Basic 4.0, binding to a data control across forms is a relatively easy task. In the past, the only way to accomplish this task was to set a **DataSourceHwnd** property to point to the **hWnd** of a data control. You are now able to set the data controls to look at one another.

To bind o a data control across forms:

1. On Form1, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the database table you want to use.
2. On Form2, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the same database table as Step 1.
3. In the Form_Load() section of Form2, add the code Set Data1.Recordset = Form1!Data1.Recordset.
4. On Form 2, create a Data Command button, setting the DataSource property to Data1.

The data controls are now bound across forms, that is, actions to the data control on either form are automatically reflected by one another.

Binding a Data DropDown to a Data Control Across Forms

Starting with Visual Basic 4.0, binding to a data control across forms is a relatively easy task. In the past, the only way to accomplish this task was to set a **DataSourceHwnd** property to point to the **hWnd** of a data control. You are now able to set the data controls to look at one another.

To bind to a data control across forms:

1. On Form1, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the database table you want to use.
2. On Form2, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the same database table as Step 1.
3. In the Form_Load() section of Form2, add the code Set Data1.Recordset = Form1!Data1.Recordset.
4. On Form 2, create a Data DropDown, setting the DataSource property to Data1.

The data controls are now bound across forms, that is, actions to the data control on either form are automatically reflected by one another.

Binding a DataOptionSet to a Data Control Across Forms

Starting with Visual Basic 4.0, binding to a data control across forms is a relatively easy task. In the past, the only way to accomplish this task was to set a **DataSourceHwnd** property to point to the **hWnd** of a data control. You are now able to set the data controls to look at one another.

To bind to a data control across forms:

1. On Form1, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the database table you want to use.
2. On Form2, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the same database table as Step 1.
3. In the Form_Load() section of Form2, add the code Set Data1.Recordset = Form1!Data1.Recordset.
4. On Form 2, create an Enhanced Data Control, setting the DataSource property to Data1.

The data controls are now bound across forms, that is, actions to the data control on either form are automatically reflected by one another.

Binding an EDC to a Data Control Across Forms

Starting with Visual Basic 4.0, binding to a data control across forms is a relatively easy task. In the past, the only way to accomplish this task was to set a **DataSourceHwnd** property to point to the **hWnd** of a data control. You are now able to set the data controls to look at one another.

To bind to a data control across forms:

1. On Form1, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the database table you want to use.
2. On Form2, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the same database table as Step 1.
3. In the Form_Load() section of Form2, add the code Set Data1.Recordset = Form1!Data1.Recordset.
4. On Form 2, create an Enhanced Data Control, setting the DataSource property to Data1.

The data controls are now bound across forms, that is, actions to the data control on either form are automatically reflected by one another.

Binding the Data Combo to a Data Control Across Forms

Starting with Visual Basic 4.0, binding to a data control across forms is a relatively easy task. In the past, the only way to accomplish this task was to set a **DataSourceHwnd** property to point to the **hWnd** of a data control. You are now able to set the data controls to look at one another. This functionality is useful for the edit portion of the Data Combo.

To bind to a data control across forms:

1. On Form1, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the database table you want to use.
2. On Form2, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the same database table as Step 1.
3. In the Form_Load() section of Form2, add the code Set Data1.Recordset = Form1!Data1.Recordset.
4. On Form 2, create a Data Combo, setting the DataSource property to Data1.

The data controls are now bound across forms, that is, actions to the data control on either form are automatically reflected by one another.

Binding to a Data Control Across Forms

Starting with Visual Basic 4.0, binding to a data control across forms is a relatively easy task. In the past, the only way to accomplish this task was to set a **DataSourceHwnd** property to point to the **hWnd** of a data control. You are now able to set the data controls to look at one another.

To bind to a data control across forms:

1. On Form1, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the database table you want to use.
2. On Form2, create a standard Data Control (Data1), setting the Database and RecordSource properties to point to the same database table as Step 1.
3. In the Form_Load() section of Form2, add the code Set Data1.Recordset = Form1!Data1.Recordset.
4. On Form 2, create a Data Grid Control, setting the DataSource property to Data1.

The data controls are now bound across forms, that is, actions to the data control on either form are automatically reflected by one another.

Bold Property Applies To

Font object
Headfont object

Bookmark Object Applies To

Bookmarks collection

BookmarkDisplay Property Applies To

SSDBData

BookmarkDisplay Property See Also

BookmarksToKeep

Bookmarks Collection Applies To

SSDBData

Bookmarks Collection See Also

Bookmark Object

BookmarksToKeep Property Applies To

SSDBData

BookmarksToKeep Property See Also

BookmarkDisplay

BorderStyle Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

BorderWidth Property Applies To

SSDBData

BorderWidth Property See Also

BevelInner
BevelOuter
BevelColorFace

BtnClick Event Applies To

SSDBGrid

Button Object Applies To

Buttons collection

ButtonEnabled Property Applies To

SSDBOptSet

ButtonFromCaption Method Applies To

SSDBOptSet

ButtonFromCaption Method See Also

ButtonFromPos

ButtonFromPos Method Applies To

SSDBOptSet

ButtonFromPos Method See Also

ButtonFromCaption method
WhereIs method

ButtonSize Property Applies To

SSDBData

ButtonVisible Property Applies To

SSDBOptSet

ButtonVisible Property See Also

Visible

Buttons Collection Applies To

SSDBOptSet

Buttons Collection See Also

Button Object

ButtonsAlways Property Applies To

Column object

ButtonsAlways Property See Also

Style

Caption Property Applies To

Button object
Column object
Group object
SSDBCommand
SSDBData
SSDBOptSet

Caption Property See Also

CaptionAlignment
Picture
PictureAlignment

CaptionAlignment Property Applies To

Group object
SSDBData
SSDBGrid
SSDBOptSet

CaptionAlignment Property See Also

PictureAlignment

Case Property Applies To

Column object

CellNavigation Property Applies To

SSDBGrid

CellNavigation Property See Also

RowNavigation

CellStyleSet Method Applies To

Column Object

CellStyleSet Method See Also

ActiveRowStyleSet
StyleSet
StyleSets collection

CellText Method Applies To

Column Object

CellText Method See Also

CellValue method

CellValue Method Applies To

Column object

CellValue Method See Also

CellText method

Change Event Applies To

SSDBCombo
SSDBGrid

Change Event See Also

For SSDBGrid

AfterUpdate
BeforeColUpdate
BeforeUpdate
BtnClick

CheckBox Column Style

The CheckBox column style can be set using the **Style** property

CheckBox3D Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Click Event Applies To

SSDBData

CloseBookmarkDropDown Event Applies To

SSDBData

CloseBookmarkDropDown Event See Also

ShowBookmarkDropDown event

CloseFindDialog Event Applies To

SSDBData

CloseUp Event Applies To

SSDBCombo
SSDBDropDown

CloseUp Event See Also

ComboDropDown event

Col Property Applies To

SSDBGrid

Col Property See Also

Grp
Row

ColContaining Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

ColMove Event Applies To

SSDBGrid

ColMove Event See Also

ColSwap event
GrpMove event
GrpSwap event

ColOffSet Property Applies To

Button object
SSDBOptSet

ColOffSet Property See Also

RowOffSet

ColPosition Method Applies To

Group object
SSDBGrid

ColPosition Method See Also

Position
GrpPosition method

ColResize Event Applies To

SSDBGrid

ColResize Event See Also

GrpResize
RowResize
SplitterMove

ColSwap Event Applies To

SSDBGrid

ColSwap Event See Also

ColMove event
GrpMove event
GrpSwap event

ColWidth Property Applies To

DataOptSet

ColorMask Property Applies To

SSDBData

ColorMask Property See Also

ColorMaskEnabled

ColorMaskEnabled Property Applies To

SSDBData

ColorMaskEnabled Property See Also

ColorMask

Cols Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid
SSDBOptSet

Cols Property See Also

Col
Row
Rows

Column Header

Column Object Applies To

Columns collection

Column Object See Also

Columns collection

ColumnHeaders Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

ColumnHeaders Property See Also

GroupHeaders

Columns Collection Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Columns Collection See Also

Column object

Columns Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Combo Box Column Style

This is different than the SSDBCombo control. Setting the Style property allows for a combo box in the grid. You can populate this box through the Grid Editor or through code.

ComboCloseUp Event Applies To

SSDBGrid

ComboCloseUp Event See Also

ComboDropDown event

ComboDropDown Event Applies To

SSDBGrid

ComboDropDown Event See Also

ComboCloseUp event

ComboDroppedDown Property Applies To

SSDBGrid

Data Widgets™ 2.0

Copyright Notice Version

What is Data Widgets?
Background on the product and its features as well as information on how to use Data Widgets with your applications.

What's New?
New features that have been added to Data Widgets 2.0.

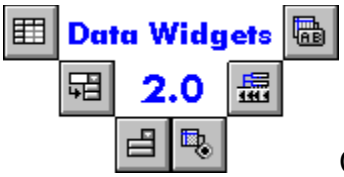
Guided Tours
Designed to get you up and running quickly by walking you through Data Widgets.

Control Descriptions
Describes each of the Data Widgets custom controls, giving detailed infomation on each.

Technical Specifications
System requirements, included files, files needed for distribution, and error messages.

Technical Support
Getting technical and product support for Sheridan products.

Control Caption Area



Control Descriptions

Data Grid
A fully editable bound grid that allows you to edit an entire record set, regardless of size, on screen without writing any code. Also supports Unbound and AddItem modes.

Data Combo
A bound combo box you can include in your application.

Data DropDown
A control used for attaching a Data Grid column to a dropdown list of values from another source of data.

DataOptionSet
Option buttons that can be used to represent field values by binding the control to a data source.

Enhanced Data Control
A front end to the Visual Basic data control adding new functionality such as bookmark navigation and page movement.

Data Command Button
Command buttons that perform database functions.

Count Property Applies To

Bookmarks collection
Buttons collection
Columns collection
Groups collection
SelBookmarks collection
StyleSets collection

Count Property See Also

Bookmarks collection
Buttons collection
Columns collection
Groups collection
StyleSets collection
Add method
Remove method
RemoveAll method

Creating Buttons at Design Time

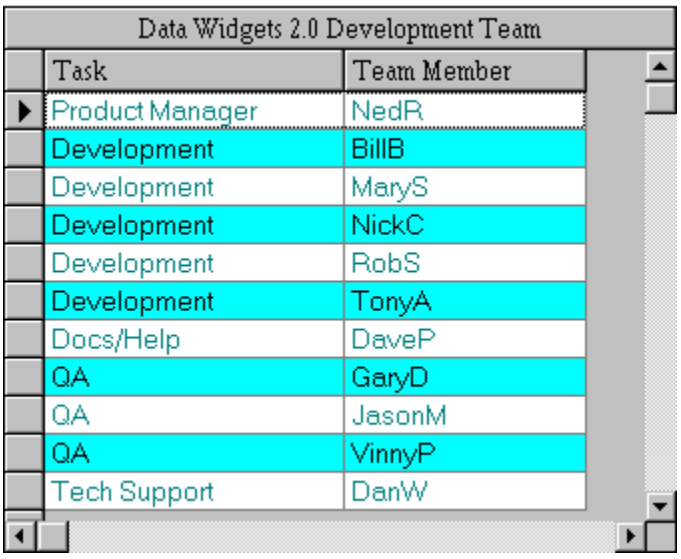
To create DataOptionSet buttons at runtime:

1. Select the total number of buttons by setting the **NumberOfButtons** property.
   The control will immediately redraw to reflect the new setting.
2. Set the **IndexSelected** property to the button number you want to modify.
   All changes made to button-specific properties will affect the button selected with this property.
3. Set the **OptionValue** property for this button.
   This is the value that will be compared against the database value.
4. Set any additional properties you wish to alter.
5. Repeat Steps 2-4 as needed.

Creating Buttons at Runtime

Creating buttons at runtime is a much simpler task thanks to the Button Object and its related Buttons Collection. The Button Object makes it possible to directly access a button without the need for selecting the **IndexSelected** property first. Additionally, you do not need to set the **NumberOfButtons** property since adding or deleting a button object within the collection automatically updates this value.

For example, to modify the fifth button's caption, you only need to write the code: SSDBOptSet1.Buttons(4).Caption = "Fifth Button".


Data Widgets 2.0 Development Team
Task
Team Member
Product Manager
NedR
Development
BillB
Development
MaryS
Development
NickC
Development
RobS
Development
TonyA
Docs/Help
DaveP
QA
GaryD
QA
JasonM
QA
VinnyP
Tech Support
DanW

Customizing the Bound Data Combo

When you select a data combo, all fields in the associated database are displayed by default. Sometimes, this is not a convenient way of displaying your data. Using the **Visible** property on individual columns allows you to display only the field you want listed.

For example, the following code displays only the first and third columns of a database with four fields:

```
Sub SSDBCombo1_InitColumnProps()
SSDBCombo1.Columns(1).Visible = False
SSDBCombo1.Columns(3).Visible = False
End Sub
```

For simplicity, you could use the Grid Editor to make the changes.

Customizing the Bound Data DropDown

When you select a Data DropDown, all fields in the associated database are displayed by default. Sometimes, this is not a convenient way of displaying your data. Using the **Visible** property on individual columns allows you to display only the field you want listed. Refer to the Properties listing for a complete listing of properties used with the Data DropDown control.

For example, the following code displays only the first and third columns of a database with four fields:

```
Sub SSDBDropDown1_InitColumnProps()
    SSDBDropDown1.Columns(1).Visible = False
    SSDBDropDown1.Columns(3).Visible = False
End Sub
```

For simplicity, you could use the Grid Editor to make the changes.

Data Combo Control

Filenames ObjectType

Anatomy of a Data Combo

The Data Combo custom control is a combo box that can be used to display/edit a field value from one record set while providing a dropdown list of field values from another set. The Data Combo functions in bound, unbound and AddItem modes.

For bound mode operation, you simply need to set four properties; two for the edit portion and two for the list portion. When you dropdown the list, it will automatically be filled with the rows and columns of the record sets you chose.

Keyboard Interface

Adding a Bound Data Combo

Customizing the Bound Data Combo

Adding an Unbound Data Combo

Achieving a 3D Look with the Data Combo

Binding to a Data Control Across Forms

Data Combo Features

The Data Combo is a bound combo box you can include in your application.

The following is a list of the features found in this control:

- Variable edit area height similar to that used in Microsoft Access
- Multiline edit area
- User is not limited to the width of the edit area for entering/displaying data
- Same formatting capabilities as the SSDBGrid control
- Full design time capabilities

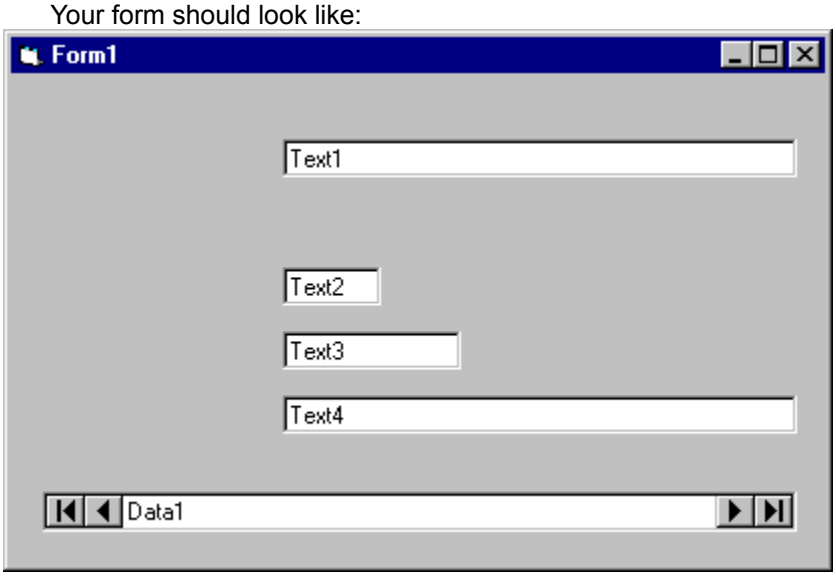
Data Combo: Exercise 1

This section guides you through the creation of a sample program using the Data Combo control. For a complete description of this control, refer to the Data Combo Control

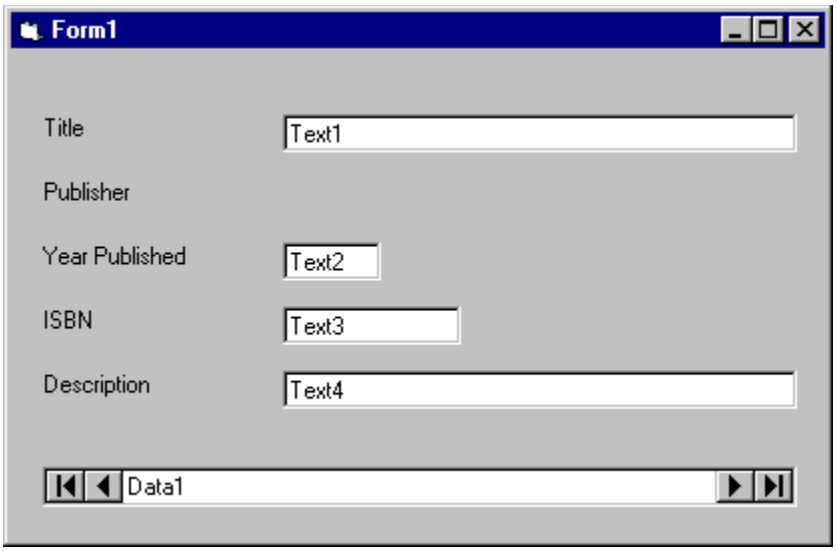
For the exercises in this section, it is assumed that you have already launched your development application (Visual Basic), and that the control has been added to your toolbox. For more information on how to do this, refer to Using Data Widgets.

In this exercise, you will create a data entry program that makes use of the Data Combo control.

1. Place a Visual Basic data control on the form.
2. Set the **DatabaseName** property to BIBLIO.MDB.
   Be sure to qualify the filename with the path of where the file is located, if needed.
3. Set the **RecordSource** property to 'Titles'.
4. Place four text boxes on the form.
   Your form should look like:

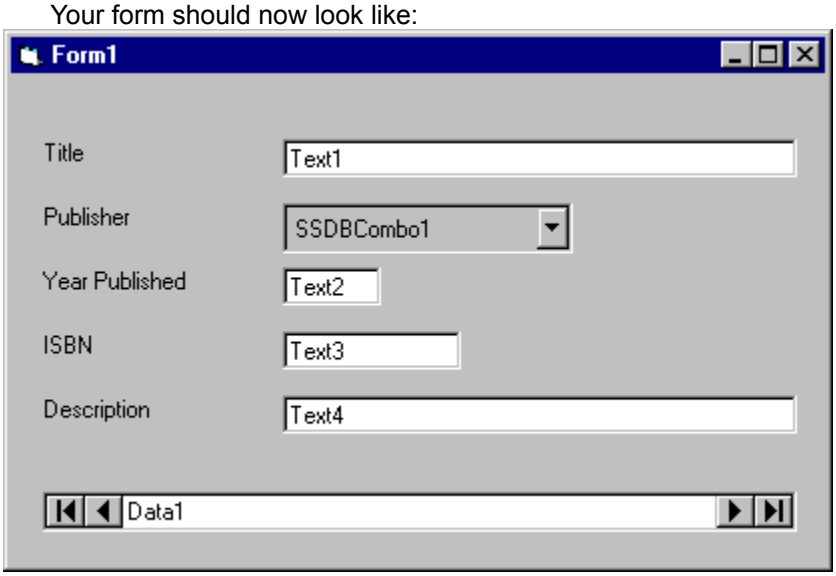


5. Create five text labels on the form.
   Your form should look like:



6. For each text box, set the **DataSource** field to *Data1* and the **DataField** as follows:
   Text1 = 'Title'
   Text2 = 'Year Published'
   Text3 = 'ISBN'
   Text4 = 'Description'

7. Add a SSDBCombo control to the form.
   Your form should now look like:



8. Place a second Visual Basic data control on the form.
9. Set the data control's **DatabaseName** property to BIBLIO.MDB.
10. Set the data control's **RecordSource** property to 'Publishers'.
11. Set the data control's **Visible** property to 'False'.
12. Set the data combo's **DataSource** property to 'Data1'.
    This binds the edit portion of the data combo to the 'Titles' table.
13. Set the data combo's **DataField** property to 'PubID'.
    This determines the field used in the edit portion of the data combo.
14. Set the data combo's **DataSourceList** property to 'Data2'.
    This binds the list portion of the data combo to the 'Publishers' table.

15. Set the data combo's **DataFieldList** property to 'PubID'.
    This determines the field used in the list portion of the data combo.

Try running your application at this point, using the data control to navigate through the records. Click on the Data DropDown and you'll see how the record in the edit field is automatically selected in the list portion. Try changing the field value by selecting another record from the list.

This is a quick example of how the Data Combo can be used in your applications. Save this project, as we'll be using it in the next exercise.

Data Combo: Exercise 2

This exercise demonstrates how you can customize the Data Combo to suit your specific needs. This exercise makes use of objects and collections. If you are not familiar with object and collections, you should refer to the Introduction to OCX Controls section before proceeding. The project used in the last exercise should be running at this point.

As you saw in the last exercise, when you click on the data combo, the entire table in the list portion displays. Sometimes this is fine, but there are times when you want to limit what is displayed and how it is displayed. The Data Combo allows you to make use of the objects that the Data Grid uses.

Let's make it so that the Data Combo only displays the fields "PubID", "Name", and "Company Name". In the **SSDBCombo1_DropDown** event, add the following code:

```
SSDBCombo1.Columns(3).Visible = False
SSDBCombo1.Columns(4).Visible = False
SSDBCombo1.Columns(5).Visible = False
SSDBCombo1.Columns(6).Visible = False
SSDBCombo1.Columns(7).Visible = False
SSDBCombo1.Columns(8).Visible = False
SSDBCombo1.Columns(9).Visible = False
```

Run your application, and drop down the Data Combo. You will now see that only the PubID, Name, and Company Name fields appear.

Altering properties is a simple task when dealing with objects. If we wanted to make the first column a different color, all we need to add in the **SSDBCombo1_DropDown** event is the following:

```
SSDBCombo1.Columns(0).Backcolor=RGB(200,200,200)
```

Try experimenting with setting different properties on the control, as well as on the columns themselves. Remember that you can also use the Grid Editor to customize the Data DropDown properties.

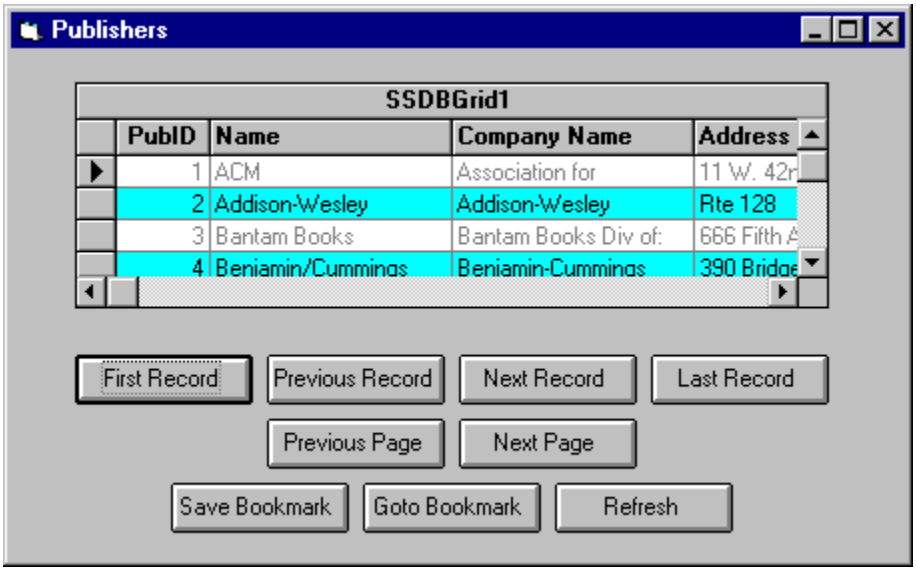
Data Command Button Control

Filenames ObjectType

The Data Command custom control is a 3D command button that can be bound to a data control. This button can be configured to automatically perform database actions as designated by you when clicked. The control also contains a speed button feature, allowing the user to hold the button in the down state to repeat a function.

A full set of appearance properties, including the capability of placing pictures on the buttons, are available with the Data Command control. The command button can be set to perform one of the following database actions:

- Goto first record
- Goto previous record
- Goto next record
- Goto last record
- Goto previous page (pages are defined by you)
- Goto next page (pages are defined by you)
- Create bookmark
- Goto bookmark
- Refresh display



Adding a Data Command Button
Speed Buttons
Binding to a Data Control Across Forms

Data Command Button Features

The Data Command Button allows you to create command buttons that perform database functions.

The following is a list of the features for this control:

- Add, Delete, Refresh, Bookmark, and Auto-Positioning functions
- Click and After Click events
- Auto-Repeat functionality
- 3D font capability
- Multiline captions
- Custom color options
- Auto-sizing capability

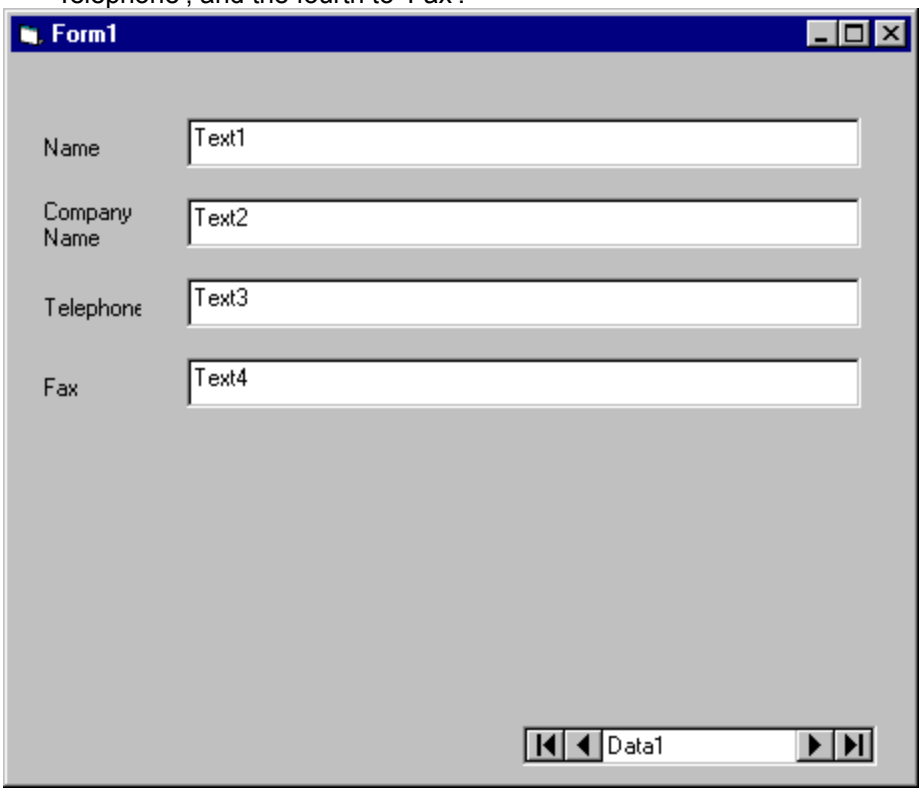
Data Command: Exercise 1

This guides you through the creation of a sample program using the Data Command control. For a complete description of this control, refer to the Data Command section.

For this exercise, it is assumed that you have already launched your development application (Visual Basic), and that the control has been added to your toolbox. For more information on how to do this, refer to Using Data Widgets.

You will create an application that makes use of the Data Command control. The database used will be the BIBLIO.MDB that ships with Visual Basic and is located in the Visual Basic home directory. If you are using an environment other than Visual Basic, and do not have the BIBLIO.MDB database, consult BIBLIO File Structure for details on the file layout.

1. Place a standard data control on the form.
2. Set the **DatabaseName** property to BIBLIO.MDB and set the **RecordSource** property to 'Publishers'. Set the **Visible** property to 'False'.
   This example will demonstrate how the Data Command buttons can replace navigation functions of the data control. We can set the data control to invisible since we will use the buttons for navigation.
3. Add four standard labels and four standard text boxes as shown below. Set the **DataSource** property to 'Data1' for all text boxes.
4. Set the **DataField** for the first text box to 'Name', set the second one to 'Company Name', the third to 'Telephone', and the fourth to 'Fax'.

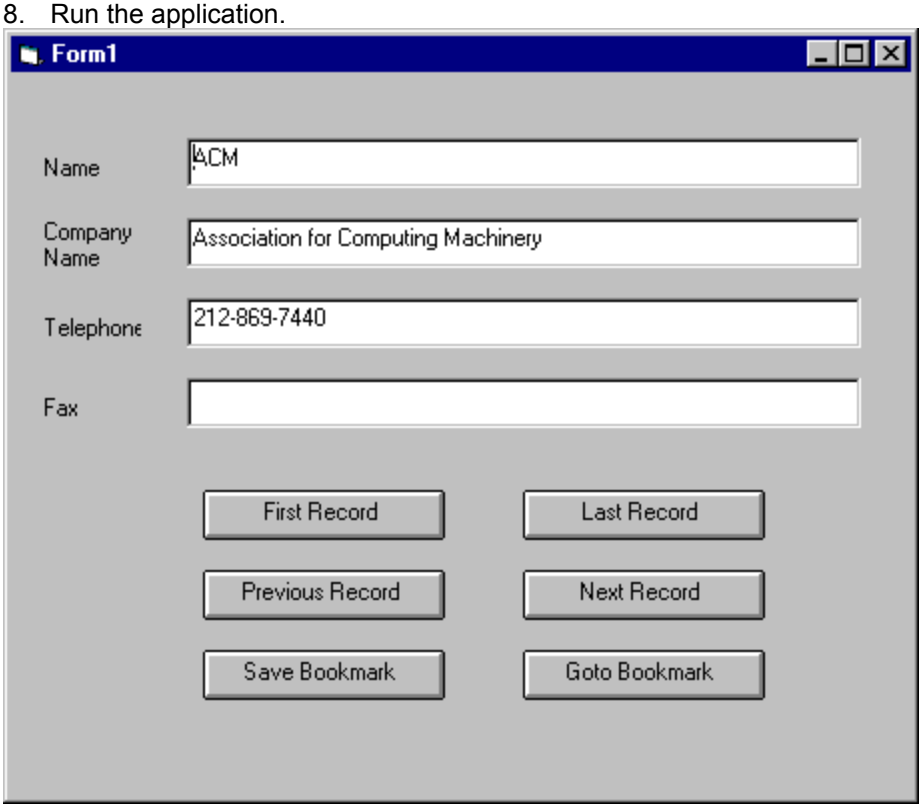


5. Add six Data Command buttons to the form setting the **DataSource** property for each one to 'Data1' and changing the **Caption** property for each one so that they look like this:

6. For each button, set the **DatabaseAction** property so it corresponds to the caption.
7. The Save Bookmark button requires one line of code to make it functional. To add it, double click on the Save Bookmark button and add the following:

```
Private Sub SSDBCommand3_AfterClick()
    SSDBCommand6.SavedBookmark = SSDBCommand3.SavedBookmark
End Sub
```

8. Run the application.



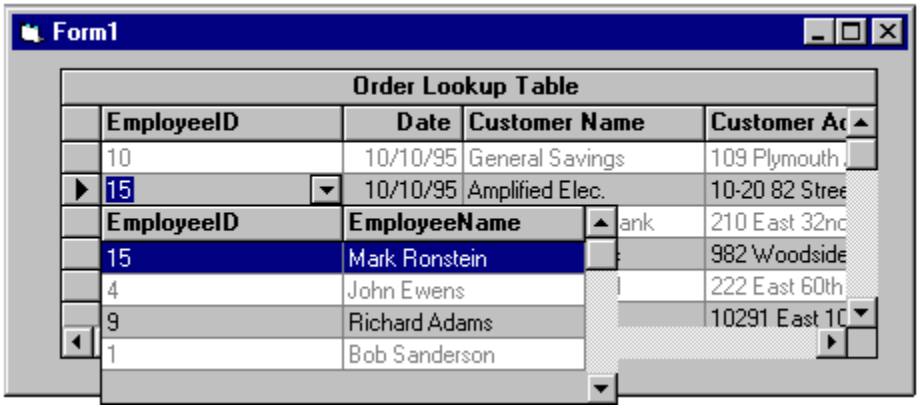
And there you have it! This is just a sampling of what the Data Command buttons are capable of. Try using this application to get a feel for how the buttons work. To see how the bookmarks work, click the Save Bookmark button, and then move to another record. Clicking on the Goto Bookmark button will take you back to the record you saved.

Data DropDown Control

Filenames ObjectType

The Data DropDown custom control is a grid that can be linked to the cells in the Data Grid (SSDBGrid) for use as a value selection list. The Data DropDown, when used in conjunction with a cell in a Data Grid, functions very similar to the Data Combo, with the exception that it does not contain an edit portion. The field that would normally be in the edit portion of a Data Combo is in the cell of the Data Grid. You can display as few or as many fields in the dropdown list as you want.

One of the advantages of the Data DropDown is that it allows you to cross-reference data to a value. Let's say you have a field (**EmployeeID)** that stores the identification number of your employees. Instead of a person needing to memorize each employee's number, you can create a Data DropDown in the **EmployeeID** field that lists the employee's full name next to their identification number in a scrollable list for easy selection. You could do the same for customers, parts, or just about any other information that you want to access from a list.



Keyboard Interface
Adding a Bound Data DropDown
Customizing the Bound Data DropDown
Adding an Unbound Data DropDown
Binding to a Data Control Across Forms

Data DropDown Features

The Data DropDown control is used for attaching a Data Grid column to a dropdown list of values from another source of data.

The following is a list of the features for this control:

- Used in conjunction with the Data Grid
- Supports multiple data modes including bound, unbound, and AddItem
- User is not limited to the width of the edit area for entering/displaying data
- Same formatting capabilities as the SSDBGrid control
- Full design time capabilities

Data DropDown: Exercise 1

This section guides you through the creation of a sample program using the Data DropDown control. For a complete description of this control, refer to the Data DropDown control.'

For the exercises in this section, it is assumed that you have already launched your development application (Visual Basic), and that the control has been added to your toolbox. For more information on how to do this, refer to Using Data Widgets.

In this exercise, you will create a simple program that makes use of the Data DropDown control.

1. Place a Visual Basic data control on the form.
2. Set the **Visible** property to **False**.
   This hides the data control when your program runs.
3. Set the **DatabaseName** property to BIBLIO.MDB.
   Be sure to qualify the filename with the path of where the file is located, if needed.
4. Set the **RecordSource** property to 'Titles'
5. Place a SSDBGrid control directly on the form by double clicking on the  tool in the Visual Basic toolbox. Resize the grid to a size that is suitable for your form.
6. For the SSDBGrid control, set the **DataSource** property to *Data1*.
   This points the Data Grid to the data control you created in Step 1.
7. For the SSDBGrid control, set the **AllowAddNew** and **AllowDelete** properties to **True**.
8. Place a second Visual Basic data control on the form.
9. For the data control, set the **Visible** property to **False**.
10. For the data control, set the **DatabaseName** property to BIBLIO.MDB.
    Be sure to qualify the filename with the path of where the file is located, if needed.
11. For the data control, set the **RecordSource** property to 'Publishers'
12. Place a Data DropDown control on the form.
    The location of the Data DropDown is unimportant since it is invisible at runtime.
13. For the SSDBDropDown control, set the **DataSource** property of the Data DropDown to the second data control.
14. For the SSDBDropDown control, set the **DataFieldList** property of the Data DropDown to 'PubID'.
15. Link the Data DropDown to the Data Grid by adding the following code in the **InitColumnProps** procedure of the Data Grid:
    SSDBGrid1.Columns(3).DropDownhWnd = SSDBDropDown1.hWnd

Try running your application at this point. Click on the PubID column, and it will drop down the 'Publisher's table. Try changing the field value by selecting another record from the list.

By default, all fields in the table display. You can selectively hide fields by adding code in the **DropDown** event of the Data DropDown.

Data Grid Control

Filenames ObjectType

Anatomy of a Data Grid

The Data Grid custom control is an editable grid that can be used to display and edit data. In just a few steps, you can have a fully functional program that allows users to view, edit, add, and delete rows in a database without a single line of code! The Data Grid can operate in bound, unbound, or AddItem mode. When working in bound mode, the Data Grid communicates with the host environment's data control, which allows your grid to interact with any database the data control is capable of using.

In AddItem mode, the Data Grid can be used as a multi-column list box, in which case, it is not linked to a database. Since the Data Grid uses virtual data management techniques, meaning it can handle any amount of data without using up all of Windows memory, you can use it to handle large lists of data. When being used in add item mode, the Data Grid stores all data in memory, which is in contrast to bound and unbound modes where only the amount of data needed to display is kept in memory.

The Data Grid is fully-customizable and can contain multiple groups and columns with the ability to specify attributes such as colors, fonts, and user permissions to individual columns and groups.

The SSDBGrid control is zero-based, which means that numbering for all rows, columns, levels, etc. start at 0. For example, the command ?SSDBGrid1.Columns(1).Caption returns the caption of the **second** column in the grid.

When using the grid in bound mode, by default, each column represents a field in the database with each column header being named after the respective database field.

Note In Data Widgets 1.0, you could not see any groups, columns or special attributes when in design mode. Version 2.0 now allows you at design time to view the grid as it will appear at runtime.

Adding a Bound Data Grid to your application

Adding an Unbound Data Grid to your application

Adding an AddItem Grid to Your Application

Grid Editor

Keyboard Interface

About SelBookmarks

Using the Data Grid as a List Box

Using the Cell Button Feature of the Data Grid

Using a Data DropDown in a Data Grid Column

Binding to a Data Control Across Forms

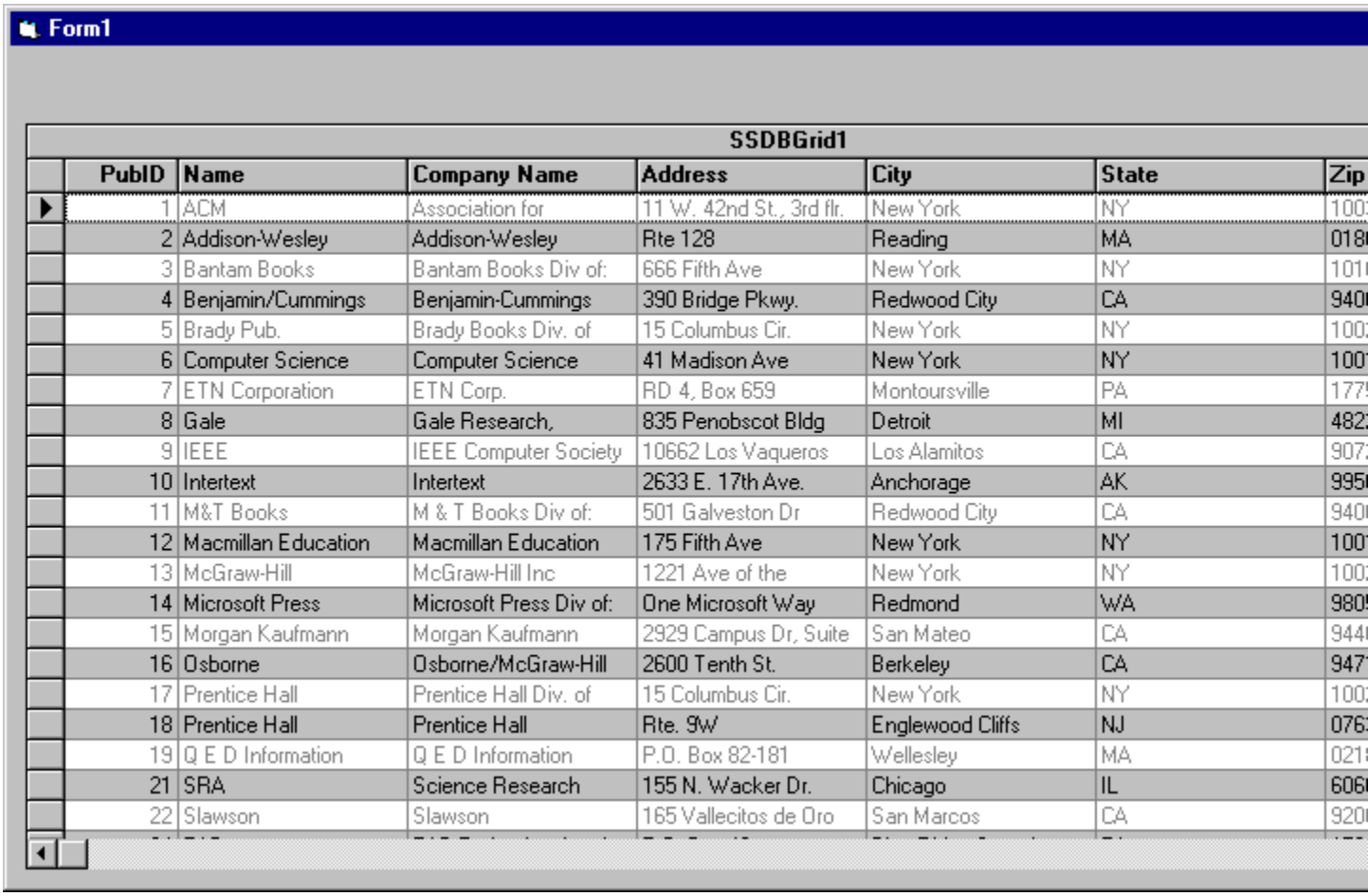
Updating Rows from a Modal Form

Data Grid Ex 1 - Part I: Adding the grid

1. Place a Visual Basic data control on the form.
2. Set the **Visible** property to **False**.
   This hides the data control when your program runs.
3. Set the **DatabaseName** property to BIBLIO.MDB.
   Be sure to qualify the filename with the path of where the file is located, if needed.
4. Set the **RecordSource** property to 'Publishers'
5. Place a SSDBGrid control directly on the form by double clicking on the  tool in the Visual Basic toolbox. Resize the grid to a size that is suitable for your form.
6. Set the **DataSource** property to *Data1*.
   This points the Data Grid to the data control you created in Step 1.
7. Set the **AllowAddNew** and **AllowDelete** properties to **True**.

Believe it or not, you just created a fully functional database grid in six steps! You can view and edit the entire table, as well as add new rows and delete existing rows.

To see for yourself, try running the application at this point (select *Start* from the **Run** menu of Visual Basic). The results should look something like:



Data Grid Ex 1 - Part II: Creating groups and columns

While the grid displays all your data, wouldn't it be nice to spice it up a little? In this next part, we're going to make use of a powerful tool called the Grid Editor. With the Grid Editor, we'll be able to create groups and columns, as well as easily specify a variety of display attributes.

With the application we created in Part I open, let's do the following:

- In Design mode, select the grid and then select "(Custom)" from the *Properties* list. The Grid Editor appears.

The first thing we want to do is divide the various fields into organizational groups:

1. Select the "Groups" tab.
   The Groups tab allows us to create and delete group definitions.
2. Click the **Add** button and type "Company Information"
3. Click the **Add** button and type "Address Information"
4. Click the **Add** button and type "Other"

You have just created three groups. Now, it's time to add some fields (herein referred to as "Columns") to the groups.

1. Select the "Columns" tab.
   The Columns tab allows us to create and delete column definitions.
2. Click on the group header labeled "Company Information".
3. Click the **Fields** button.
   The Fields button allows us to automatically create columns from a bound database.
4. Select the fields "Name" and "Company Name" from the *Fields Selection* list.
5. Click the **OK** button.
   You have just added these two fields to the "Company Information" group.
6. Click on the group header labeled "Address Information".
7. Click the **Fields** button.
8. Select the fields "Address", "City", "State", and "Zip" from the *Fields Selection* list.
9. Click the **OK** button.
   You have just added these four fields to the "Address Information" group.
10. Click on the group header labeled "Other".
11. Select the fields "Telephone", "Fax", and "Comments" from the *Fields Selection* list.
12. Click the **OK** button.
    You have just added these four fields to the "Other" group.

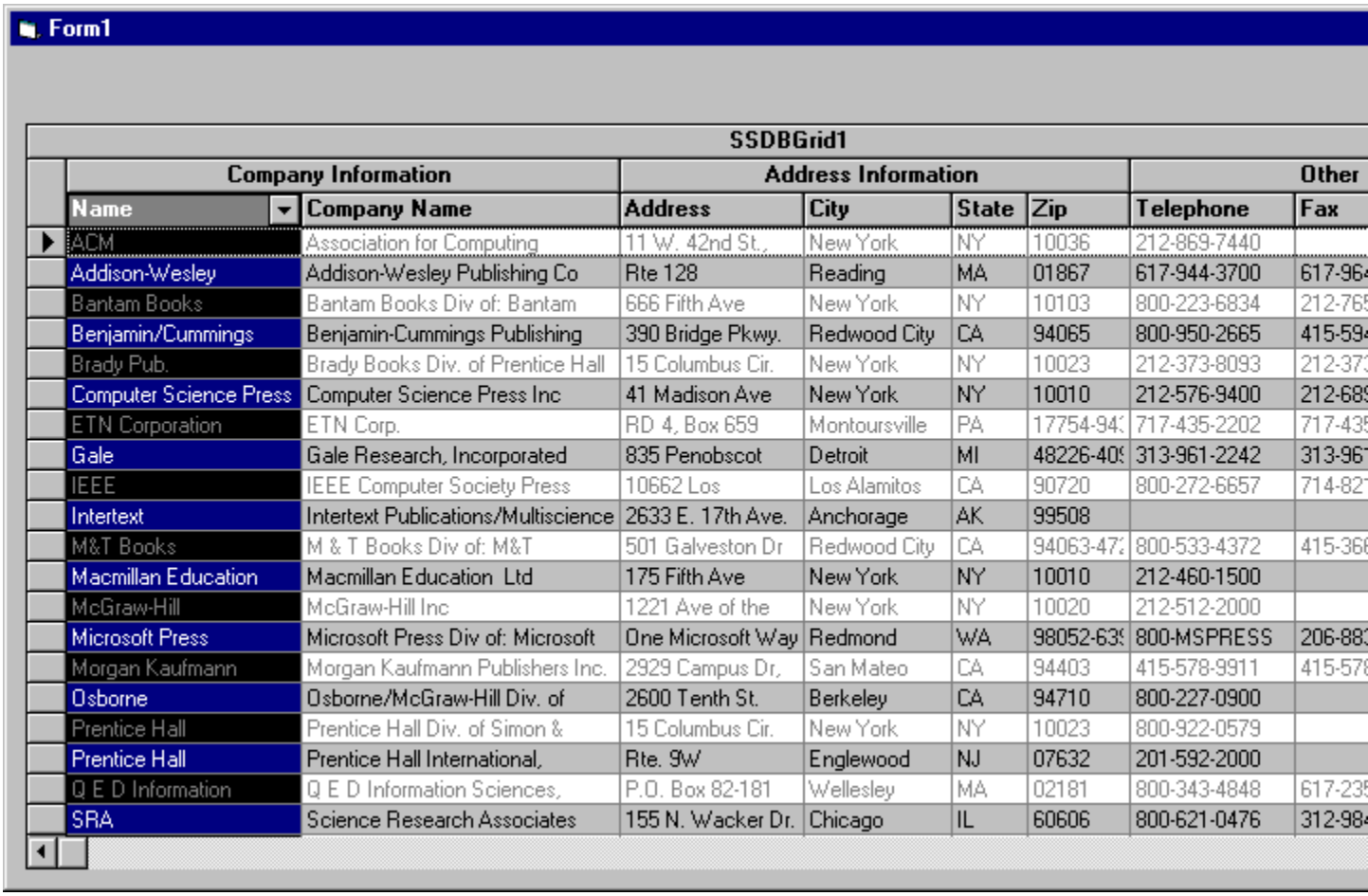
You've just created a grid layout making use of groups and columns! Resize the groups to your liking by clicking on the right edge of the group headers and dragging them to either the left for smaller, or right for larger. You can do the same for the columns by clicking on the column headers and dragging. You'll notice that we left two fields out of our grid, "PubID" and "Comments". The Grid Editor allows you to selectively use fields in your grid.

Once you have specified your layout, it's a good idea to actually *Apply* it to the grid by clicking the **Apply** button. This updates the SSDBGrid control with the layout you just designed. You'll then want to close

the Grid Editor for now by clicking the **OK** button.

Note You can apply your changes *and* close the Grid Editor simply by clicking the **OK** button. If your changes have not yet been applied, you will be prompted to apply them prior to closing.

If you want, try running the application at this point (select *Start* from the **Run** menu of Visual Basic). The results should look something like:



Data Grid Ex 1 - Part III: Customizing the Data Grid

The Data Grid is quite versatile when it comes to customizing both look and functionality. The Grid Editor allows you to work with a number of properties at design time. In this next section, we're going to customize our grid so that you can begin to see the wide-range of possibilities the grid offers.

At this point, we want to go back into the grid editor so that we can customize our grid. If you don't remember, we can launch the grid editor by selecting the grid control then select "(Custom)" from the *Properties* list.

Changes made in the Grid Editor will not take affect in the real grid until we click the **Apply** button in the Grid Editor.

Let's begin to customize our grid:

Customizing General tab options

The **General** tab is a tree-structure representing the various properties that can be set for the Data Grid. The General tab is the first tab to appear when you launch the Grid Editor. When you select a property for modification, options for that property appear to the right. The two exceptions to this are the (Add Items...) and StyleSets options, which are both fully explained in 'Chapter 11 - The Data Grid Control'.

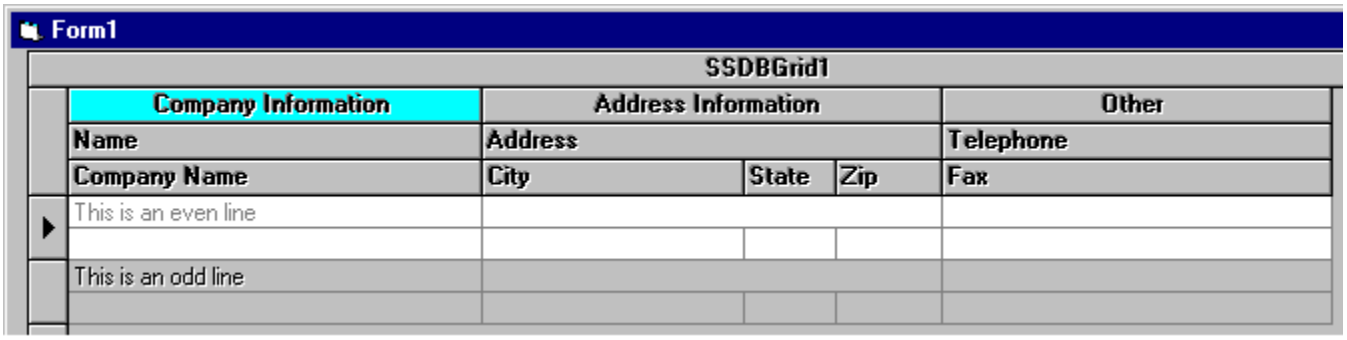
Set the following properties as shown:

Property	Value	What It Does
Caption	"Publisher Database"	Specifies the caption title for the Data Grid
CaptionAlignment	"0 - Left Justify"	Left justifies the caption
AllowAddNew	True	Allows users to add new records to the Data Grid
AllowDelete	True	Allows users to delete records from the Data Grid
Font	"Arial" for Name 8 for Point Size	Specifies the font name and size to be used for the grid text.
HeadFont3D	"Inset w/light shading"	Gives the text of your grid headers a 3D appearance.
HeadFont	"Arial" for Name 12 for Point Size	Specifies the font name and size to be used for the grid headers, including caption.
AllowColumnMoving	"2 - Anywhere"	Allows users to move columns anywhere on the grid
AllowColumnSwapping	"2 - Anywhere"	Allows users to swap columns anywhere on the grid
DividerType	"Horizontal"	Determines what type of row divider is used.
SelectByCell	True	Allows the selection of an entire row if the user clicks on a cell.

GroupHeadLines	2	Allows the Group Headers to occupy two rows.
LevelCount	2	Allows each record to occupy two rows. You will need to decide what columns you want on each level.

Customizing Group options

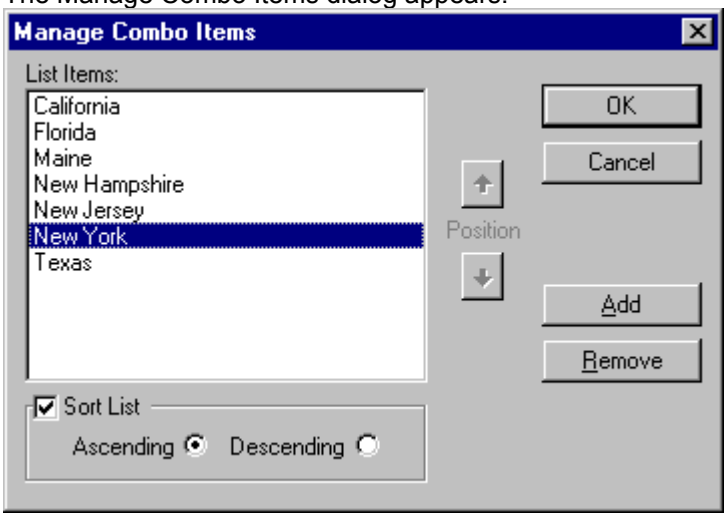
1. Select the "Groups" tab.
2. Select the group "Other" from the **Name** list.
3. Set the **AllowSizing** property to False.
   This prevents users from changing the width of the "Other" group.
4. Select the group "Company Information" from the **Name** list.
5. Set the **HeadBackColor** property to the color that you want to use.
   The color you choose can either be from the palette shown, a custom color you define by clicking on the **Custom Color** list, or a pre-defined system color from the **System Color** list that corresponds to your Windows color scheme.
6. Repeat Step 5 for as many groups as you want to define colors for.
7. Move the "Company Name", "City", "State", and "Zip" columns so that they appear on the second level.
   To move a column from one level to another, select the column header and drag it to the appropriate level. The results will look something like:



Customizing Column options

1. Select the "Columns" tab.
2. Select the column "Name" from the **Name** list.
3. Set the **Case** property to '2 - UPPER'.
   This causes all fields in the "Name" column to appear in uppercase.
4. Select the column "State" from the **Name** list.
5. Set the **Style** property to '3 - Combo Box'.
   This causes the State field to appear as a combo box, allowing the user to choose from a list of entries. When you select **Combo Box**, the **Setup** button appears, allowing you to modify the contents of the combo box.

6. Click the **Setup** button.
   The Manage Combo Items dialog appears:



7. Add the names of states listed in the graphic above by clicking the **Add** button and then typing the name in the *Add List Item* edit box.
8. Select **Sorted** by clicking in the check box.

This will sort your entries in ascending or descending order based on your preference. You can manually sort the list by selecting fields individually and clicking the **Up** or **Down** buttons.

9. Click the **OK** button.

You've just completed modifying the grid layout! Remember, in order for the changes we specify in the Grid Editor to take affect on the actual grid, you must click the **Apply** button to activate the changes. We're done with the Grid Editor, so you can click the **OK** button to go back to the form.

Data Grid Ex 1 - Part IV: Running the application

You can now run your application to see how the changes have affected your grid. Some items to note about the grid are:

- Notice how the Company Name field displays all entries in uppercase.
- Try clicking on the State column. A combo box should appear listing the states you entered.
- Try resizing groups and columns by selecting the right side of the column or group header and dragging left or right. Try doing this for the "Other" group. Remember that you disabled the resizing for this group.

The Grid Editor can be run again at anytime in the future, that is, you can make further changes to your grid whenever you're in design mode.

The exercise just completed is just a taste of what's possible with the Data Grid. The best way to learn about the grid is to experiment with different settings. The Grid Editor provides context-sensitive help throughout should you have any questions about a specific property. You should also refer to Chapter 11 for an in-depth view of the Data Grid control.

Data Grid Features

The Data Grid is a fully editable bound grid that allows you to edit an entire record set, regardless of size, on screen without writing any code.

The following is a list of the features found in the Data Grid control:

- Functionally and visually consistent with data grids in Microsoft Access and Visual Basic 4.0
- Support for movable groups and columns
- Optional dropdowns in headings allow users to select from a list of available fields and/or groups at runtime
- Additional cell types include checkbox, button, label, and combo box
- Multiline row formats
- Pictures and text in cells and headings
- Use of fonts and colors by column, row, and cell
- Drag and Drop of cells
- Supports multiple data modes including bound, unbound, and AddItem.
- AddItem at design time
- Supports Sheridan Style Sets

Data Grid: Exercise 1 (Bound Mode)

This section guides you through the creation of some sample programs using the Data Grid control. For a complete description of this control, refer to the Data Grid Control.

For this exercise, it is assumed that you have already launched your development application (Visual Basic), and that the control has been added to your toolbox. For more information on how to do this, refer to Using Data Widgets.

In this exercise, you will create an application that makes use of the Data Grid control. The database used will be the BIBLIO.MDB that ships with Visual Basic and is located in the Visual Basic home directory. If you are using an environment other than Visual Basic, and do not have the BIBLIO.MDB database, consult BIBLIO File Structure for details on the file layout.

Data Grid: Exercise 2 (Unbound Mode)

In this exercise, you will create a fully functional unbound Data Grid control. The database used will be the UNBOUND.MDB that resides in the \SAMPLES\CHAP5 directory of Data Widgets 2.0. The reason this is an unbound data grid is because we are not using the Visual Basic data control, and are using our own routines to deal with data.

Note: This exercise makes use of the Microsoft DAO 2.5 Object Library by using Visual Basic commands such as OpenDatabase and OpenRecordset. If you do not have this library available, VB may generate a "User-defined type not defined" message when running this application. If you encounter this situation, simply enable this library through the "References..." dialog under Visual Basic's *Tools* menu.

---

1. Place a SSDBGrid control directly on the form by double clicking on the  tool in the Visual Basic toolbox. Resize the grid to a size that is suitable for your form.
2. Set the **DataMode** property to'1 - Unbound'.
3. Set the **AllowAddNew** and **AllowDelete** properties to 'True'
4. Add the following code to the **(declarations)** section of your form:

```
Dim db As Database
Dim rs As Recordset
Dim ct As Integer

' DB represents the database to be used
' RS represents the recordset to be used
' CT represents the count of fields in the table
```

5. Add the following code to the **Form_Load** procedure:

```
Dim i As Integer
    Set db = OpenDatabase("unbound.mdb")
    Set rs = db.OpenRecordset("Titles")
    ct = rs.Fields.Count

    SSDBGrid1.Columns.RemoveAll

    For i = 0 To ct - 1
      SSDBGrid1.Columns.Add i
      SSDBGrid1.Columns(i).Visible = True
      SSDBGrid1.Columns(i).Caption = rs.Fields(i).Name
    Next i
```

In this section, the database is assigned, a table selected, and the number of fields in the table is given to CT. The RemoveAll is issued to ensure that there are no existing columns when the grid goes to add columns from the database. In the 'For i = 0 To ct - 1' loop, columns are created based on the number of fields in the database, and headers are assigned based on the field name.

6. Add the following code to the **SSDBGrid1_UnboundReadData** event:

```
Dim r, i, j As Integer

If IsNull (StartLocation) Then
    If ReadPriorRows Then
            rs.MoveLast
    Else
            rs.MoveFirst
    End If
```

```
Else
        rs.Bookmark = StartLocation
        If ReadPriorRows Then
                rs.MovePrevious
        Else
                rs.MoveNext
        End If

End If

For i = 0 to Rowbuf.RowCount - 1
        if rs.BOF or rs.EOF Then Exit For

        For j = 0 to ct - 1
                Rowbuf.Value(i,j) = rs(j)
        Next j

        Rowbuf.Bookmark(i) = rs.Bookmark

        If ReadPriorRows Then
                rs.MovePrevious
        Else
                rs.MoveNext
        EndIf

r = r + 1

Next i

Rowbuf.RowCount = r
```

The UnboundReadData event reads data into the grid. In this section, the program first looks to see if it's dealing with an empty grid or not, by checking *StartLocation*. If it is empty, then the first or last record is gone to based on the *ReadPriorRows* value (which indicates if scrolling should occur forwards or backwards). If it is not empty, the database bookmark is set to the start location and we move backwards or forwards based on *ReadPriorRows.*
We then read in the data, only stopping when we reach the beginning or end the file or when we reach *RowCount* - 1. As we add each row, we keep track of a counter r, which we set *RowBuf.RowCount* to when we finish.

7. Add the following code to the **SSDBGrid1_UnboundAddData** event:

```
Dim i As Integer

rs.AddNew

For i = 0 to (ct-1)
        If Not IsNull (Rowbuf.Value (0,i)) Then
                rs(i) = Rowbuf.Value(0,i)
        End If
Next i

rs.Update
rs.MoveLast
NewRowBookmark = rs.Bookmark
```

The **UnboundAddData** event adds a new row. In this section, the database is prepared for a row addition, and the field values are passed from the ssRowBuffer to the database for each field. The database is then updated. Finally, we move to the last row in the database, and point the

NewRowBookmark.

8. Add the following code to the **SSDBGrid1_UnboundDeleteRow** event:

```
rs.Bookmark = Bookmark
rs.Delete
```

The **UnboundDeleteRow** event deletes a row. In this section, the database is pointed to the row being deleted, and then actually deletes it.

9. Add the following code to the **SSDBGrid1_UnboundWriteData** event:

```
Dim I As Integer

rs.Bookmark = WriteLocation

rs.Edit

For i = 0 To (ct-1)
    If Not IsNull(RowBuf.Value(0,i)) Then
        rs(i) = RowBuf.Value(0,i)
    End If
Next i

rs.Update
```

The **UnboundWriteData** event writes a row that has been changed. In this section, the database is pointed to the row being written, and then put into edit mode. The data is then passed from the grid to the database for each field, and then issued an update command.

10. Run your application.

Data Grid: Exercise 3 (AddItem Mode)

In this exercise, you will create a fully functional AddItem Data Grid control.

1. Place a SSDBGrid control directly on the form by double clicking on the  tool in the Visual Basic toolbox. Resize the grid to a size that is suitable for your form.
2. Set the **DataMode** property to '2 - AddItem'.
3. Set the **Cols** property to 3.

This tells the grid that we will use three columns.

4. Set the **FieldDelimiter** property to ! (exclamation mark).

This property determines the character that represents the start and end of a field.

5. Set the **FieldSeparator** property to , (comma).
   This property determines the character that represents a separation between fields.
6. Add the following code to the **SSDBGrid1_InitColumnProps** event:

```
Dim I As Integer
For I = 0 to 20
        SSDBGrid1.AddItem "!Hello!,!There!,!World!"
Next I
```

Run your program. You'll see that it functions just like an Unbound or Bound grid.

Select a Data Widget control:

Data Grid
Data Combo
Data DropDown
DataOptionSet
Enhanced Data
Data Command Button

DataField Property Applies To

Column object
SSDBCombo
SSDBCommand
SSDBData
SSDBOptSet

DataField Property See Also

DataSource

DataFieldList Property Applies To

SSDBCombo
SSDBDropDown

DataFieldList Property See Also

DataMode
DataSource
DataSourceList

DataMode Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

DataOptionSet Control

Filenames ObjectType

The DataOptionSet custom control allows you to use 3D options buttons that can be bound to a database field. Using the DataOptionSet control, you can easily incorporate option buttons for use in your database application. Instead of entering data which can lead to typographical errors, users may simply click on an option button to select a pre-defined value for a field. For example, using one DataOptionSet, you could have four option buttons to represent various credit-card payment methods.

If the value of the active field equals the value set for the control, the option button will be automatically clicked. If another DataOptionSet button matches a value in the field or there is no match, the button will automatically be clicked off.

One control can contain multiple option buttons, which can be added (up to 100 total) or deleted, as specified by you at either design or runtime. Appearance of the buttons are user-controllable through the use of layout properties. The DataOptionSet is zero based, meaning that numbering of buttons starts at zero.

The DataOptionSet makes use of objects and collections.

Keyboard Interface

Adding the DataOptionSet
Creating Buttons at Design Time
Creating Buttons at Runtime
Binding to a Data Control Across Forms

DataOptionSet Features

The DataOptionSet control allows the binding of data fields to option buttons for representation of field values. For example, if you were writing a point-of-sale system, you could have four option buttons, each representing various credit cards ("Visa", "MasterCard", "American Express", and "Discover"). Clicking on the appropriate option button automatically changes the value in the database and allows you to store the type of payment that was used.

The following is a list of the features for this control:

- Multiline captions
- One control creates unlimited option buttons that are bound to the same data field
- Saves Windows resources
- Automatic and manual row/column positioning
- Each individual button can have a caption with optional picture
- Custom color options
- 3D capability

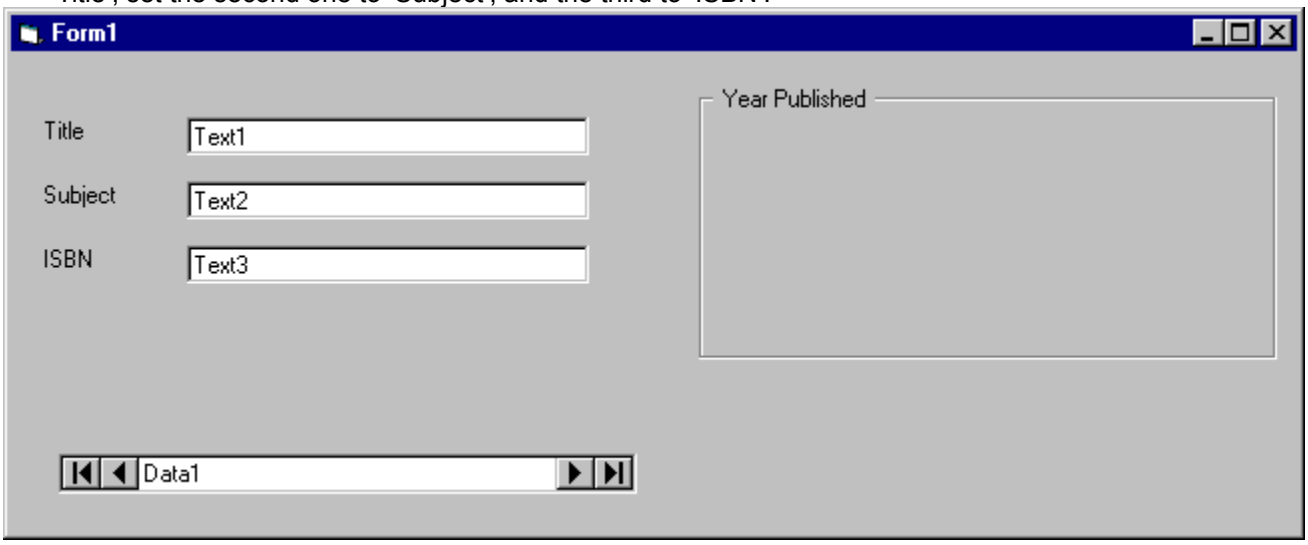
DataOptionSet: Exercise 1

This section guides you through the creation of a sample program using the DataOptionSet control. For a complete description of this control, refer to the DataOptionSet.

For this exercise, it is assumed that you have already launched your development application (Visual Basic), and that the control has been added to your toolbox. For more information on how to do this, refer to Using Data Widgets.

In this exercise, you will create an application that makes use of the DataOptionSet control. The database used will be the BIBLIO.MDB that ships with Visual Basic and is located in the Visual Basic home directory. If you are using an environment other than Visual Basic, and do not have the BIBLIO.MDB database, consult BIBLIO File Structure for details on the file layout.

1. Place a standard data control on the form.
2. Set the **DatabaseName** property to BIBLIO.MDB and set the **RecordSource** property to 'Titles'.
3. Add three standard labels, three standard text boxes and a standard frame as shown below. Set the **DataSource** property to 'Data1' for all three text boxes. Set the **DataField** for the first text box to 'Title', set the second one to 'Subject', and the third to 'ISBN'.



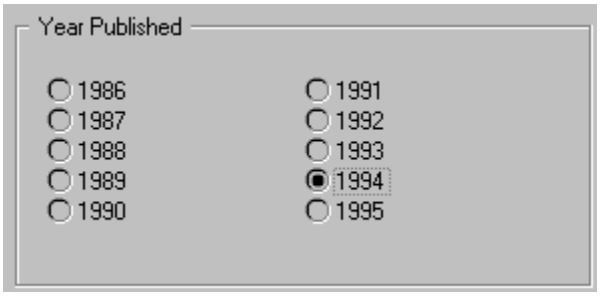
4. Add a DataOptionSet control to the form, within the frame captioned 'Year Published'.
5. For the SSDBOptSet control, set the **DataSource** property to Data1 and the **DataField** property to 'Year Published'.

This binds the control to the 'Year Published' field of the database specified in Data1.

6. Also for the SSDBOptSet control, set the **NumberOfButtons** property to 10 and the **Cols** property to 2.

This creates ten option set buttons on the form, divided into two columns.

7. Set the **Caption** property to "1986".
   This changes the caption for the first button to 1986.
8. Set the **IndexSelected** property to 1.
   This changes the selected button from the first (0) to the second (1). The DataOptionSet is zero-based. When you first create a DataOptionSet, the first button is automatically selected.
9. Repeat Steps 7 and 8 until you have renamed all 10 buttons to look like:



10. For each button, set the **OptionValue** property to match its caption.
    Remember, you must first set the **IndexSelected** property to specify the button you want to work with. The **OptionValue** property compares against the value in the database.
11. Run your application.

Try moving through some of the records using the data control. Notice how the option buttons change as you move. You'll notice that some records cause the DataOptionSet to not select any button, this is because the date does not match any of the option values specified. Remember, you didn't need to write a single line of code! However, it is possible to accomplish what we just did through code. The following procedure can be used in place of Steps 6 through 10:

```
Private Sub Form_Load()
        SSDBOptSet1.NumberOfButtons = 10
        SSDBOptSet1.Cols = 2

        SSDBOptSet1.Buttons(0).Caption = "1986 "
        SSDBOptSet1.Buttons(0).OptionValue = "1986"

        SSDBOptSet1.Buttons(1).Caption = "1987 "
        SSDBOptSet1.Buttons(1).OptionValue = "1987"

        SSDBOptSet1.Buttons(2).Caption = "1988 "
        SSDBOptSet1.Buttons(2).OptionValue = "1988"

        SSDBOptSet1.Buttons(3).Caption = "1989 "
        SSDBOptSet1.Buttons(3).OptionValue = "1989"

        SSDBOptSet1.Buttons(4).Caption = "1990 "
        SSDBOptSet1.Buttons(4).OptionValue = "1990"

        SSDBOptSet1.Buttons(5).Caption = "1991 "
        SSDBOptSet1.Buttons(5).OptionValue = "1991"

        SSDBOptSet1.Buttons(6).Caption = "1992 "
        SSDBOptSet1.Buttons(6).OptionValue = "1992"

        SSDBOptSet1.Buttons(7).Caption = "1993 "
        SSDBOptSet1.Buttons(7).OptionValue = "1993"

        SSDBOptSet1.Buttons(8).Caption = "1994 "
        SSDBOptSet1.Buttons(8).OptionValue = "1994"

        SSDBOptSet1.Buttons(9).Caption = "1995 "
        SSDBOptSet1.Buttons(9).OptionValue = "1995"
End Sub
```

While the code is rather lengthy, you can begin to see how easy it is to work with DataOptionSet buttons through code.

DataSourceList Property Applies To

SSDBCombo

DataSourceList Property See Also

DataFieldList
DataMode
DataSource

DataType Property Applies To

Column object

DatabaseAction Property Applies To

SSDBCommand

DefColWidth Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

DelayInitial Property Applies To

SSDBCommand
SSDBData

DelayInitial Property See Also

DelaySubsequent

DelaySubsequent Property Applies To

SSDBCommand
SSDBData

DelaySubsequent Property See Also

DelayInitial

Distributing Your Application

Once you have created a program using Data Widgets controls, you must distribute the OCX files with your application. There are no separate design time and runtime versions of the controls, therefore, the same OCX files you develop with can be shipped with your application.

Filename	Description
SSDATA16.OCX	16-Bit OCX containing SSDBData, SSDBOptSet, SSDBCommand
SSDATA32.OCX	32-Bit OCX containing SSDBData, SSDBOptSet, SSDBCommand
SSDATB16.OCX	16-Bit OCX containing SSDBGrid, SSDBDropDown, SSDBCombo
SSDATB32.OCX	32-Bit OCX containing SSDBGrid, SSDBDropDown, SSDBCombo

Support files needed for distribution

Due to the nature of the OLE architecture, the new OCX controls require that a number of supporting files be shipped with your application. They have been installed in your system directory and are:

dao2516.dll	vaen2.dll
msajt200.dll	vaen21.olb
msjeterr.dll	vb40016.dll
msjetint.dll	vb4en16.dll
oc25.dll	vbajet.dll
typelib.dll	vbdb16.dll

A note about OLE file distribution

The introduction of OCX controls has created some confusion amongst developers when it comes to the five files used for OLE (COMPOBJ.DLL, OLE2.DLL, OLE2DISP.DLL, OLE2NLS.DLL, STORAGE.DLL). **These files are only needed to support 16-Bit applications created with Data Widgets.**

Windows 95 and Windows NT

If your application is running under Windows 95 or Windows NT, then the OLE DLLs are part of the OS and you do not need to install or update these files, provided the version numbers match below:

Windows 95	
compobj.dll	Version 2.2
ole2.dll	Version 2.2
ole2disp.dll	Version 2.1
ole2nls.dll	Version 2.1
storage.dll	Version 2.2

Windows NT	
compobj.dll	Version 2.1

ole2.dll	Version 2.1
ole2disp.dll	Version 2.1
ole2nls.dll	Version 2.1
storage.dll	Version 2.1

Windows 3.x and Windows for Workgroups 3.x

If your application is running under Windows 3.x or Windows For Workgroups 3.x, you **must** make sure that these DLLs are installed for any applications created with the OCX version of Data Widgets. These DLLs are included with the Data Widgets installation disks and are copied when you install Data Widgets under either of these environments.

Windows 3.x and Windows for Workgroups 3.x	
compobj.dll	Version 2.03
ole2.dll	Version 2.03
ole2disp.dll	Version 2.03
ole2nls.dll	Version 2.03
storage.dll	Version 2.03

DividerStyle Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

DividerStyle Property See Also

DividerType

DividerType Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

DividerType Property See Also

DividerStyle

DoClick Method Applies To

SSDBCombo
SSDBCommand
SSDBDropDown
SSDBGrid

DropDown Event Applies To

SSDBCombo
SSDBDropDown

DropDown Event See Also

CloseUp event

DropDownHwnd Property

Applies To

Description

Specifies the handle of the Data DropDown (Hwnd property) to be linked with the Data Grid.

Syntax

object.**DropDownHwnd**[= *hwnd*]

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
hwnd	A variant expression specifying the handle of the DataDropDown to be linked.

Remarks

Set this property to the Hwnd property of a Data DropDown to enable a link between the grid and Data DropDown. The Data Grid will automatically display the dropdown button in a cell when it is active.

This property is only available at runtime.

The **DropDownHwnd** property can only be used with the Data DropDown control. It will **not** work with other controls.

Example

The following code ties the second column of a Data Grid to a Data DropDown:

```
SSDBGrid1.Columns(1).DropDownHwnd = SSDBDropDown1.Hwnd
```

DropDownHwnd Property Applies To

Column object

DroppedDown Property Applies To

SSDBCombo
SSDBData
SSDBDropDown

DroppedDown Property See Also

CloseBookmarkDropdown event
ShowBookmarkDropdown event

Enhanced Data Control

Filenames ObjectType

The Enhanced Data Control (EDC) is an enhanced version of the Data Control that ships with Visual Basic. The EDC is used in conjunction with the data control rather than taking its place. The EDC can be oriented either horizontally or vertically, and can be sized to your liking at design time.

Some of the enhancements that the EDC provides include bookmark storage allowing you to return to a particular row at a later time, next page and previous page buttons, the ability to selectively enable/disable features of the EDC, and a speed button feature allowing the user to hold down a button to repeat its function.

Adding the Enhanced Data Control
Anatomy of the Enhanced Data Control
Finding Information with the Enhanced Data Control
What are Bookmarks?
Speed Buttons
Binding to a Data Control Across Forms

Enhanced Data Control: Exercise 1

This section guides you through the creation of a sample program using the Enhanced Data control. For a complete description of this control, refer to the Enhanced Data Control section.

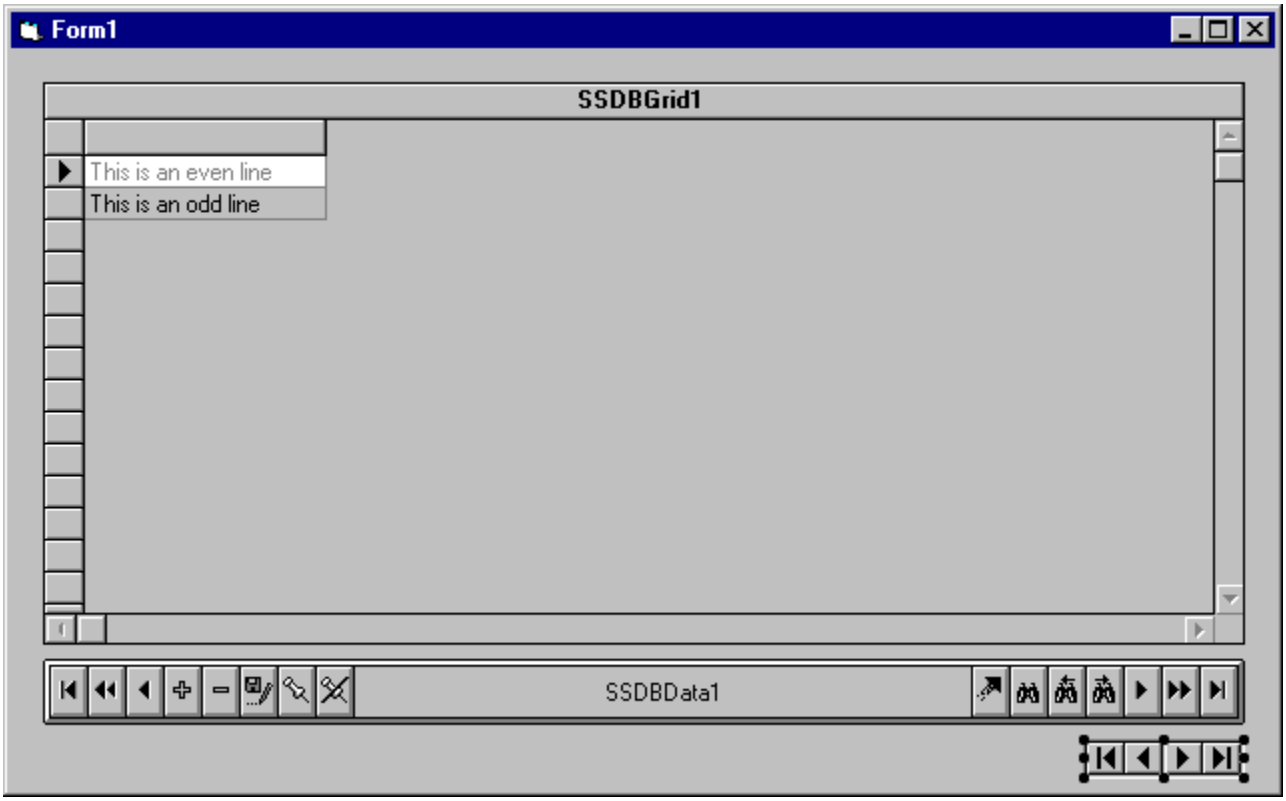
For these exercises, it is assumed that you have already launched your development application (Visual Basic), and that the control has been added to your toolbox. For more information on how to do this, refer to Using Data Widgets.

You will create an application that makes use of the Enhanced Data control. The database used will be the BIBLIO.MDB that ships with Visual Basic and is located in the Visual Basic home directory. If you are using an environment other than Visual Basic, and do not have the BIBLIO.MDB database, consult BIBLIO File Structure for details on the file layout.

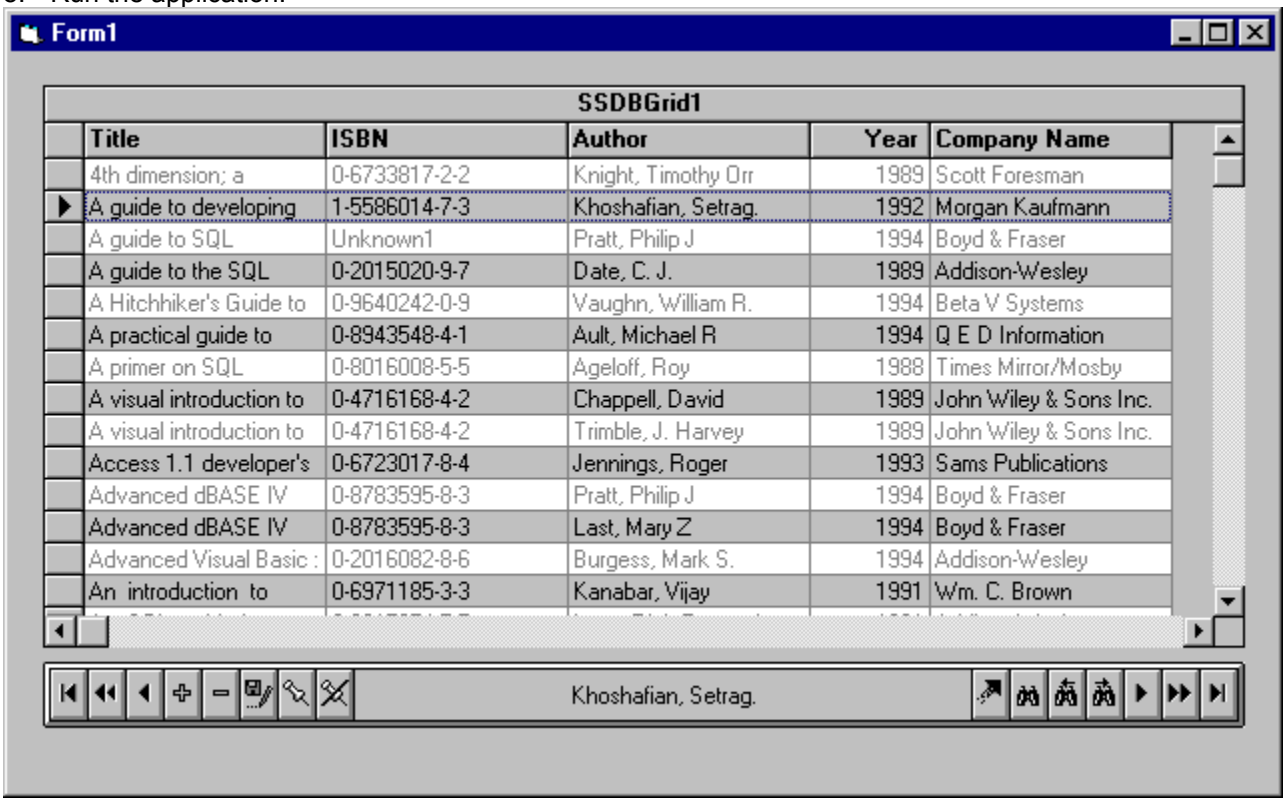
1. Place a standard data control on the form.
2. Set the **DatabaseName** property to 'BIBLIO.MDB' and set the **RecordSource** property to 'All Titles'. Set the **Visible** property to 'False'.
   This example will demonstrate how the Enhanced Data control supplements the standard data control by adding new navigational features. The standard Data Control is required for access to the database, but is not required for navigation.
3. Place a SSDBGrid control directly on the form by double clicking on the  tool in the Visual Basic toolbox.

   Resize the grid to a size that is suitable for your form.
4. Set the grid's **DataSource** property to 'Data1'
5. Place an Enhanced Data Control (EDC) directly on the form by double clicking on the  tool in the Visual Basic toolbox.

   Resize the EDC so that it fits the width of the form.



6. Set the EDC's **DataSource** property to 'Data1' and the **DataField** property to 'Author'.
   This binds the EDC to the database. Remember that the EDC relies on the standard data control for access to the database. However, you can make the standard data control invisible so that your users do not see it at runtime.
7. Set the grid's **AllowAddNew** and **AllowDelete** properties to **True**.
   This allows you to add and delete records within the database by clicking the corresponding buttons on the EDC.
8. Run the application.



Try scrolling through the application.

Use the following navigational buttons to help you:

First Record

Jumps to the first record in the database.

Previous Page

Jumps to the previous page in the database. A page is determined by the setting of **PageValue**.

Previous Record

Jumps to the previous record in the database.

Next Record

Jumps to the next record in the database.

Next Page

Jumps to the next page in the database. A page is determined by the setting of **PageValue**.

Last Record

Jumps to the last record in the database.

Try adding and deleting records (you might want to make a backup copy of the database before you do this).

Use the following buttons to help you:

Add Record

Adds a new record to the end of the database.

Delete Record

Deletes the selected record from the database.

Update Record

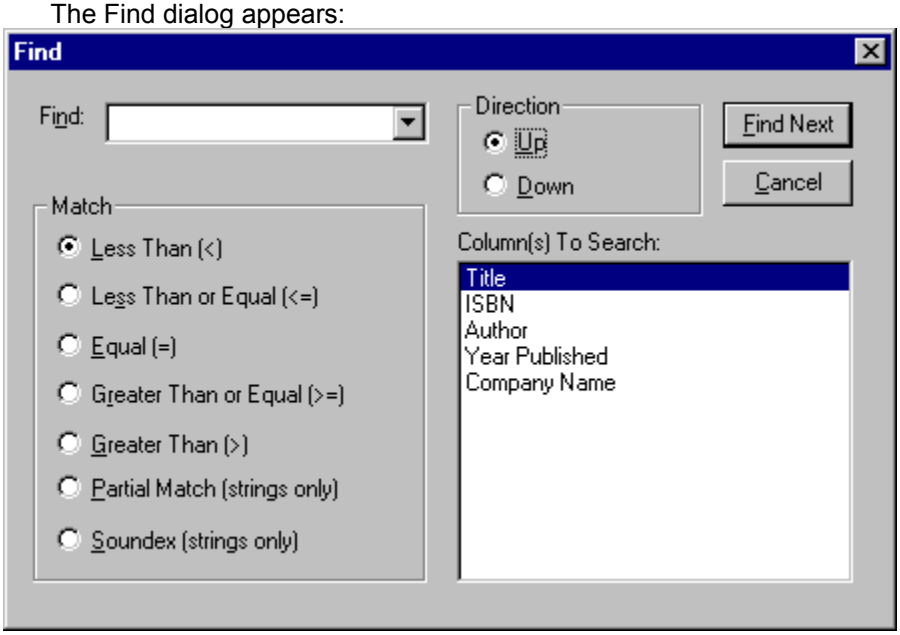
Updates the selected record in the database.

Don’t delete this project, we’ll need it in the next exercise.

Enhanced Data Control: Exercise 2

One of the many useful features of the Enhanced Data Control is its searching capabilities. This exercise demonstrates some of the capabilities of the EDC's search routine. The project used in the last exercise should be running at this point.

1. To activate the Find dialog, click the  button.
   The Find dialog appears:



2. In the **Find** box, type 'John Wiley & Sons Inc.'.
3. Select 'Equal (=)' for the type of match.
4. Select 'Company Name' for the search column.
5. Click **Find Next**
   The first occurrence of this text will be found.
6. To find the next occurrence of this text, click the **Find Next**  button.
7. To find the last occurrence of this text, click the **Find Previous**  button.

The **Find** dialog can be a very powerful tool for locating information in large databases. It also has advanced searching capabilities that include Soundex searching and Partial String searching. To try Soundex searching, replace the text in Step 2 with "Jon". To try Partial String searching, replace the text in Step 2 with "John"

Enhanced Data Control: Exercise 3

Bookmarks are a powerful tool that allow you to "flag" a record in a database and later go back to it at the click of the mouse. This exercise demonstrates how you can use bookmarks to better manage your database. The project used in the last exercise should be running at this point.

Before we start, it is important to understand that using bookmarks is a two-step process. The first step is to Add the bookmark, the second step is to Goto the bookmark. You can only go to a bookmark that you have added.

1. Scroll though the database and select the first record you want to add a bookmark for.
2. Click the **Add Bookmark**  button.
   You will notice that two buttons (the Delete Bookmark and Goto Bookmark buttons) just became un-grayed and available for selection.
3. Scroll though the database and select another record you want to add a bookmark for.
4. Click the **Add Bookmark**  button.
5. Scroll through the database some more, and click the **Goto Bookmark**  button.
6. Select the bookmark you want to go to from the dropdown.
   Notice how the bookmark is represented by the field you are bound to. This is because bookmarks are binary and can not be displayed, so a text string must be associated with it. If you use the **Add** method to add bookmarks in code, you need to include the second parameter which is the display string.
7. Click the **Clear All Bookmarks**  button.
   All bookmarks that you created have just been deleted.

Now you can begin to see the power of the Enhanced Data Control. The three exercises we performed are a sampling of what you can do. The EDC also allows you to customize its button interface including the bitmap itself as well as selectively toggle buttons on or off.

Enhanced Data Control Features

The Enhanced Data Control behaves as a front end to the Visual Basic data control adding new functionality such as bookmark navigation and page movement.

The following is a list of the features for this control:

- Re-position recordset by selecting from a dropdown list of up to 100 previously marked rows
- Buttons that perform database actions such as add, delete, update, plus bi-directional *n*th record paging
- Store and sort multiple bookmarks in the bookmark dropdown list
- Conditional and Soundex searching
- Optional user-defined pictures for each button
- Auto-repeat movement keys (forward, backward) plus bidirectional *n*th record paging
- 3D Font capability
- Caption text rotation
- Display/Hide any button
- Custom events give full programmatic control over button clicks and navigation.

Error Messages

The following is a list of trappable errors that could occur at runtime when using the Data Widgets custom controls.

Error Number	Description
30422	ssItemNotInList **"Item is not in list"** Validation determined that the current edit text is not in the associated dropdown list.
30423	ssNullEdit **"A null value is not allowed for this field"** Validation determined that the current edit text is null, but nulls are not allowed.
30424	ssBadTypeEdit **"The value is not of the correct type for this field"** Validation determined that the data type of the value currently in edit text does not match the data type of the associated list.
30425	ssBitMap **"PictureDropDown must be a bitmap"** You tried setting the PictureDropDown property to something other than a bitmap
30426	ssDropItemsRange **"Value must be from 1 to 100"** You tried setting the MaxDropDownItems or MinDropDownItems outside the allowed range.
30427	ssMaxLessMinDropItem **"MaxDropDownItems cannot be less than MinDropDownItems"** You tried setting MaxDropDownItems to a number smaller than the MinDropDownItems value.
30428	ssMinMoreMaxDropItem **"MinDropDownItems cannot be greater than MaxDropDownItems"** You tried setting the MinDropDownItems to a value greater than the MaxDropDownItems.
30434	ssBadParam **"Invalid parameter"** You tried passing an invalid parameter to a method.
30430	ssValue0to10 **"Value must be from 0 to 10"** You tried setting a property outside the allowed range.
30433	ssBadHost

	"Host environment does not support formatting"
	You tried setting the BevelWidth property outside the allowed range.
30435	ssLevelCount
	"LevelCount must be from 1 to 10"
	You tried setting a property outside the allowed range.
30436	ssFieldDelim
	"FieldDelimiter can be any one character or the word 'None'"
	You tried setting FieldDelimeter to something other than that allowed.
30437	ssFieldSep
	"FieldSeparator can be any one character, the word 'Tab', or 'Space'"
	You tried setting FieldSeperator to something other than that allowed.
30438	ssLevelnoGroup
	"The level count cannot be greater than zero if there are no groups"
	You tried setting the property to something other than that allowed.
32033	ssEDCBevelInner
	"BevelInner must be from 0 to 2"
	You tried setting the property to something outside the allowed range.
32034	ssEDCBevelOuter
	"BevelOuter must be from 0 to 2"
	You tried setting the property to something outside the allowed range.
32035	ssEDCBevelWidth
	"BevelWidth must be from 0 to 10"
	You tried setting the property to something outside the allowed range.
32036	ssEDCBorderStyle
	"BorderStyle must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32037	ssEDCBorderWidth
	"BorderWidth must be from 0 to 10"
	You tried setting the property to something outside the allowed range.
32038	ssEDCRoundedCorners
	"RoundedCorners must be either 0 or 1"
	You tried setting the property to something outside the allowed range.

32039	ssEDCBevelColorScheme
	"BevelColorScheme must be from 0 to 2"
	You tried setting the property to something outside the allowed range.
32040	ssButtonSize
	"ButtonSize must be from 5 to 100"
	You tried setting the property to something outside the allowed range.
32041	ssPictureButtons
	"PictureButtons must be of type Bitmap"
	You tried specifying a type other than Bitmap for PictureButtons.
32042	ssShowBookmarkButtons
	"ShowBookmarkButtons must be from 0 to 7"
	You tried setting the property to something outside the allowed range.
32043	ssShowFirstLastButtons
	"ShowFirstLastButtons must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32044	ssShowPageButtons
	"ShowPageButtons must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32045	ssShowPrevNextButtons
	"ShowPrevNextButtons must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32046	ssShowAddButton
	"ShowAddButton must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32047	ssShowCancelButton
	"ShowCancelButton must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32048	ssShowDeleteButton
	"ShowDeleteButton must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32049	ssShowUpdateButton
	"ShowUpdateButton must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32050	ssShowFindButtons

	"ShowFindButtons must be from 0 to 3"
	You tried setting the property to something outside the allowed range.
32051	ssEDCAlignment
	"Alignment must be from 0 to 8"
	You tried setting the property to something outside the allowed range.
32052	ssBookmarkDisplay
	"BookmarkDisplay must be from 0 to 2"
	You tried setting the property to something outside the allowed range.
32053	ssBookmarksToKeep
	"BookmarksToKeep must be from 1 to 100"
	You tried setting the property to something outside the allowed range.
32054	ssEDCCaptionAlignment
	"CaptionAlignment must be from 0 to 2"
	You tried setting the property to something outside the allowed range.
32055	ssColorMaskEnabled
	"ColorMastEnabled must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32056	ssPageValue
	"PageValue must be from 2 to 1000"
	You tried setting the property to something outside the allowed range.
32057	ssPictureCaptionAlignment
	"PictureCaptionAlignment must be from 0 to 14"
	You tried setting the property to something outside the allowed range.
32058	ssFindBufferSize
	"FindBufferSize must be from 1 to 1000"
	You tried setting the property to something outside the allowed range.
32059	ssOrientation
	"Orientation must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32060	ssEDCBalloonHelp
	"BalloonHelp must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32062	ssDeleteRecord
	"Choose Yes to delete the current row or No to exit"

	Prompts user before deleting a row.
32063	ssNoBookmarksLeft
	"There are no more bookmarks to remove"
	You tried removing a bookmark when there are none.
32064	ssCantMove
	"No Current Record"
	You've tried moving to where there is no record.
32065	ssRemoveLastButton
	"Cannot remove last button"
	You've attempted to remove the last button.
32066	ssOptionNotUnique
	"Option value must be unique"
	You've tried setting the option value to something that is already an option value for another button.
32067	ssAlignment
	"Alignment must be either 0 or 1"
	You tried setting the property to something outside the allowed range.
32068	ssPictureAlignment
	"PictureAlignment must be from 0 to 3"
	You tried setting the property to something outside the allowed range.
32069	ssBevelColorScheme
	"BevelColorScheme must be from 0 to 2"
	You tried setting the property to something outside the allowed range.
32070	ssCols
	"Cols must be from 1 to 10, but not greater than the total number of buttons."
	You tried setting the property to something outside the allowed range.
32071	ssColWidth
	"ColWidth must be a positive integer"
	You tried setting the property to something outside the allowed range.
32072	ssFont3D
	"Font3D must be from 0 to 4"
	You tried setting the property to something outside the allowed range.
32073	ssHeightGap
	"HeightGap must be from 1 to 1000"
	You tried setting the property to something outside the allowed range.
32074	ssIndexSelected

	"IndexSelected must be from 0 to the total number of buttons - 1"
	You tried setting the property to something outside the allowed range.
32075	ssMinColWidth
	"MinColWidth must be a positive integer equal to or greater than 30"
	You tried setting the property to something outside the allowed range.
32076	ssMinheight
	"MinHeight must be a positive integer equal to or greater than 15"
	You tried setting the property to something outside the allowed range.
32077	ssNumberOfButtons
	"NumberOfButtons must be from 0 to 99"
	You tried setting the property to something outside the allowed range.
32078	ssPictureMetaHeight
	"PictureMetaHeight must be a positive integer"
	You tried setting the property to something outside the allowed range.
32079	ssPictureMetaWidth
	"PictureMetaWidth must be a positive integer"
	You tried setting the property to something outside the allowed range.
32080	ssWidthGap
	"WidthGap must be a positive integer"
	You tried setting the property to something outside the allowed range.
32097	ssDCBBevelWidth
	"BevelWidth must be from 0 to 10"
	You tried setting the property to something outside the allowed range.
32098	ssDCBDelayValue
	"Delay Value must be from 1 to 5000"
	You tried setting the property to something outside the allowed range.
32099	ssDCBPageValue
	"PageValue must be from 2 to 1000"
	You tried setting the property to something outside the allowed range.
32100	ssDCBBorderStyle
	"BorderStyle must be either 0 or 1"
	You tried setting the property to something outside the

	allowed range.
32101	ssDCBCaptionAlignment **"CaptionAlignment must be from 0 to 8"** You tried setting the property to something outside the allowed range.
32102	ssDCBPictureAlignment **"PictureAlignment must be from 0 to 14"** You tried setting the property to something outside the allowed range.
32103	ssDCBDatabaseAction **"DatabaseAction must be from 0 to 8"** You tried setting the property to something outside the allowed range.

Even Row (Row 0)

FieldDelimiter Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

FieldDelimiter Property See Also

FieldSeparator

FieldLen Property Applies To

Column object

FieldSeparator Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

FieldSeparator Property See Also

FieldDelimiter

FieldValue Property

Applies To

Description

Returns the field value for the active record.

Syntax

object.**FieldValue**[= *number*]

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
number	An integer expression specifying the field value.

Remarks

FieldValue contains the value of the field specified in **DataField** for the active record.

Example

The following example displays a message box containing the employee name corresponding to the current record, assuming that **DataField** is set to the field "EmployeeName":

```
MsgBox ("Current Employee: "+SSDBGrid1.FieldValue)
```

FieldValue Property Applies To

SSDBData

FindBufferSize Property Applies To

SSDBData

FindBufferSize Property See Also

FindDialog
ShowFindButtons

FindDialog Property Applies To

SSDBData

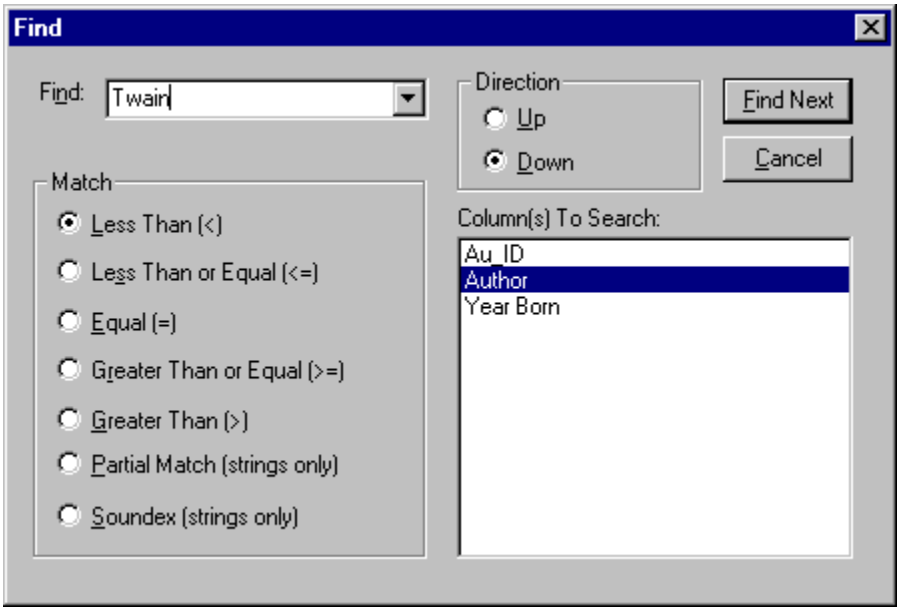
FindDialog Property See Also

FindBufferSize
ShowFindButtons

Finding Information with the Enhanced Data Control

Finding information in a database field is a snap with the Enhanced Data Control.

To activate the Find dialog, click the  button. You can use the  and  buttons to continue searching once you've found a match.



Find	Specify the data to search for. A list of recent searches is available by clicking the  button.
Direction	Specifies the direction in which to search. Selecting **Down** will search from the current point to the end of the database. Selecting **Up** will search from the current point to the start of the database.
Less Than	Match only if the text entered in the *Find* dialog is less than the value in the database. Examples of this are 1 < 2 and APPLE < BEAR
Less Than or Equal To	Match only if the text entered in the *Find* dialog is less than or equal the value in the database. Examples of this are 2 <= 2 and APPLE < BEAR
Equal To	Match only if the text entered in the *Find* dialog equals the value in the database. Examples of this are 5 = 5 and DOG = DOG
Greater Than or Equal	Match only if the text entered in the *Find* dialog equals or exceeds the value in the database. Examples of this are 7 >= 2 and DOT >= DOS
Greater Than	Match only if the text entered in the *Find* dialog exceeds the value in the database. Examples of this are 10 > 9 and TREE > BARK.
Partial Match	Match only if a portion of the string specified in the *Find* dialog matches a portion in the database. An example of

this is specifying "Eng" in the *Find* dialog and returning "Engine" and "England". This works for strings only.

Soundex — Match only if the string sounds like one in the database. An example of this is specifying "Skool" and returning "School". This works for strings only.

FirstRow Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Font Object Applies To

SSDBCombo
SSDBCommand
SSDBData
SSDBDropDown
SSDBGrid
SSDBOptSet

Font Object See Also

Bold
Font3D
Italic
Name
Size
Strikethrough
Underline

Font3D Property Applies To

SSDBCombo
SSDBCommand
SSDBData
SSDBDropDown
SSDBGrid
SSDBOptSet

Font3D Property See Also

Caption

Fonts

Fonts are supported through the **Font** object. At design time fonts are set through one of the font properties (For example: **Font**). Depending on the development environment you are using, a dialog box containing font information may be available so that you can set properties of the **Font** object. If not, you can set the font properties through the Property Pages. The following properties are supported by the **Font** object:

Properties

Bold	Size	Underline
Italic	StrikeThrough	
Name		

Fonts can be set either through the font dialog at design time or by setting properties of the **Font** object at runtime.

ForeColorEven Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

ForeColorEven Property See Also

ForeColor
ForeColorOdd

ForeColorOdd Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

ForeColorOdd Property See Also

ForeColor
ForeColorEven

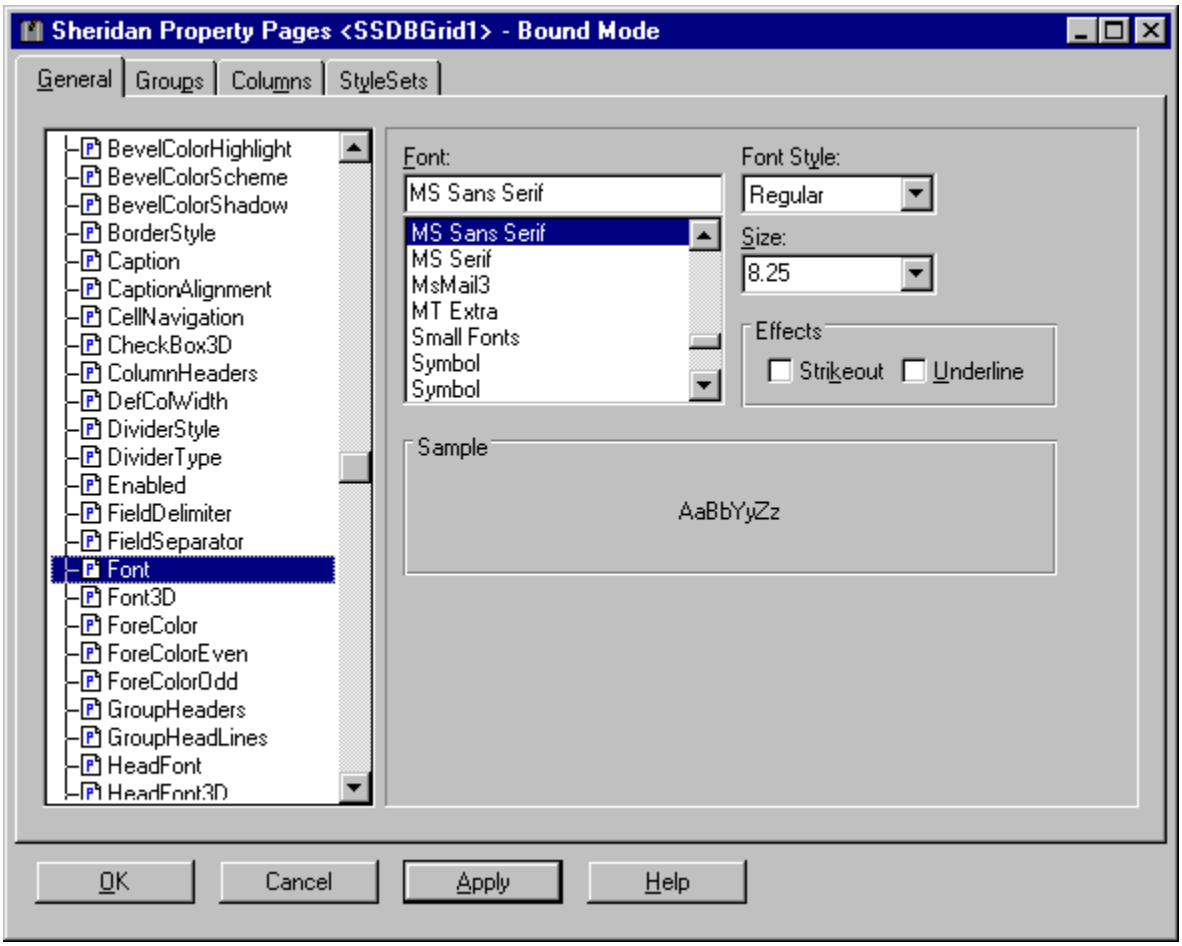
GetBookmark Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Grid Editor

The grid editor is used to easily customize the appearance of the Data Grid, Data Combo, and Data DropDown controls. Through a tabbed dialog, you can define the number of columns and groups as well as their appearance and associated properties.

The Grid Editor uses a work area called the Design Grid that simulates how your Data Grid will appear. The Design Grid works in much the same way as the Data Grid with the ability to move, resize, and swap columns and groups.



Grid Editor: Accessing

The Grid Editor is activated by selecting the **(Custom)** property from the properties list, or by selecting "Properties" from the right-click menu of Visual Basic. You can use the Grid Editor at any time in design time to make changes to your grid. The Grid Editor simulates layout by displaying a grid known as the Design Grid.

As a demonstration, the Grid Editor (GRIDEDIT.EXE) can be executed as a standalone program. In this case, you will be able to select **Open** from the *File* menu and select a database to use. All functions of the grid can be explored.

Clicking the **OK** button applies the changes you have made and exits the Grid Editor.

Clicking the **Apply** button applies the changes you have made and remains in the Grid Editor.

Clicking **Cancel** aborts the changes you have made and exits the Grid Editor.

Grid Editor: Columns tab

The Columns Tab allows you to define the columns that apepar in your grid. Columns appear in the Design Grid, allowing you to visualize how your grid will look at runtime.

Resizing

The width of the grid or the selected column can be changed by entering a value (in twips) in the text boxes labeled "Grid Width" and "Column Width".

Alternatively, you can resize the width of the grid by dragging the splitter, and you can resize the width of a column by clicking on the right-edge of its header and dragging the column to the desired size.

Adding a column to the Design Grid

1. Click the **Add** button.
2. Specify the name for the column in the "Add Column" dialog.
   The column will be added to the grid.

Removing a column from the Design Grid

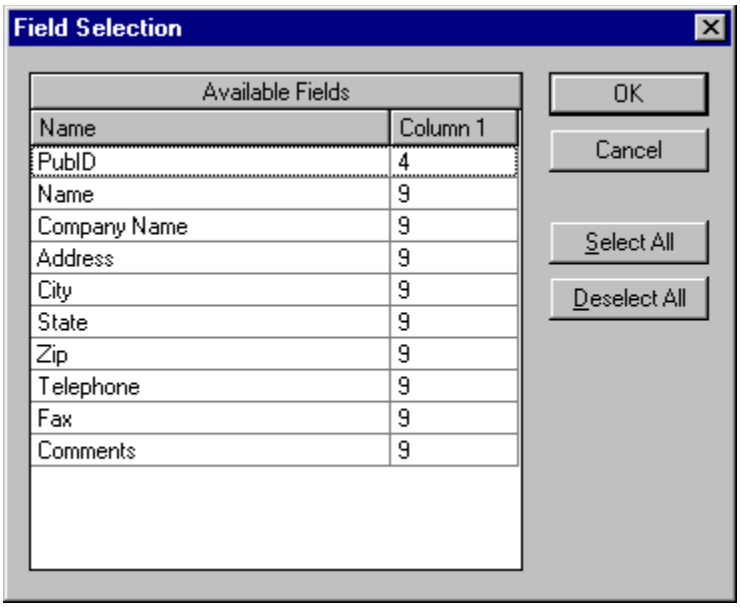
1. In the Design Grid, click the header of the column to remove.
2. Click the **Remove** button.

Note: It is possible to remove multiple columns at once. Simply click on each header corresponding to the column you want removed.

Adding columns to the Design Grid from a bound datasource

It is possible to automatically create columns based on the field structure of a database that the grid is bound to.It is possible to automatically create columns based on the field structure of a database that the grid is bound to. To add columns from a bound datasource:

1. Click the **Fields** button.
   The "Field Selection" dialog appears listing all fields in the database.



2. Select the fields you want to appear as columns.
   To select all fields in the database, click the **Select All** button.
3. Click the **OK** button.
   The selected fields appear as columns in the Design Grid.

Grid Editor: General tab

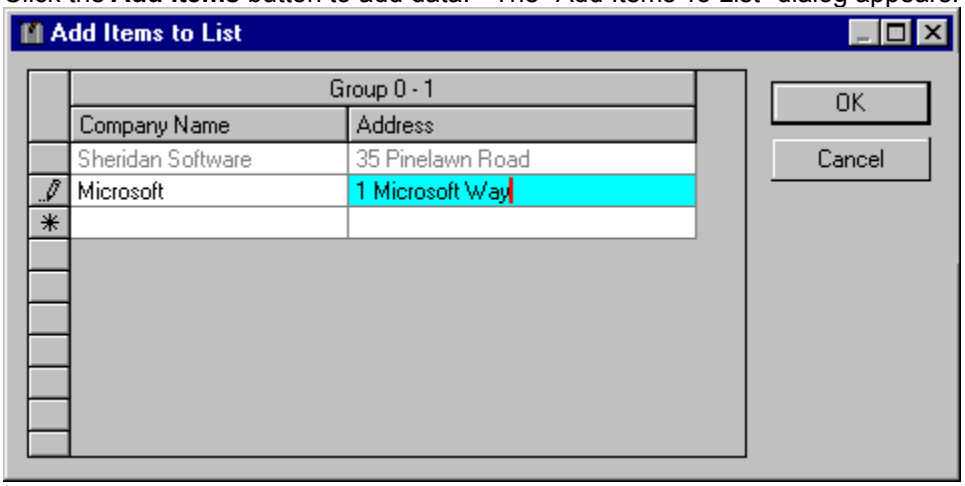
The General tab has the appearance of a standard Sheridan Property Page. Through a tree structure, you are able to select and modify properties that apply to the grid as a whole. To modify a property, simply select it from the tree and make the desired changes from the options presented on the right.

There are two items on the General Tab that need special explanation; (Add Items...) and StyleSets.

Adding items to an AddItem grid

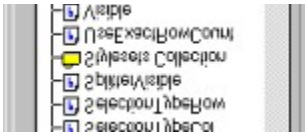
If **DataMode** is set to AddItem, the (Add Items...) option appears on the top of the tree. Selecting this option allows you to manually fill an AddItem grid with data.

Click the **Add Items** button to add data. The "Add Items To List" dialog appears:



Fill in the data as needed, clicking the **OK** button when you are finished. Clicking the **Cancel** button allows you to exit without saving your changes.

Working with StyleSets



The Grid Editor allows you to easily maintain and apply StyleSets. If you are not familiar with StyleSets, you should first read about StyleSets. Before you can apply a StyleSet, you must first define it. You can have an unlimited amount of StyleSets for any given grid, however, StyleSets are not interchangable between grids.

When you first access the Grid Editor's *General* Tab, the StyleSets Collection will be collapsed by default. To expand the collection, double-click it. This will display any StyleSets that have already been created:

To see the individual properties applicable to the StyleSet, double-click that StyleSet:

To modify a property within a StyleSet, simply select it from the tree and make the desired changes from the options presented on the right.

Adding StyleSets

1. Select StyleSets from the tree structure.
2. Click the **Add** button that appears to the right.
3. Specify a name in the "Add StyleSet" dialog.
4. The StyleSet now appears in the tree:

Removing (Deleting) StyleSets

1. Select the StyleSet to remove from the tree structure.
2. Click the **Remove** button that appears to the right.
   The selected StyleSet is deleted.

Applying StyleSets takes place in the *StyleSets* tab.

Grid Editor: Groups tab

The Groups Tab allows you to define the groups that appear in your grid. Groups allow you to logically arrange fields that are associated with one another. For example, you could have a group called "Address Information" that contains the Address, City, State, and Zip Code fields from a database. Groups appear in the Design Grid, allowing you to visualize how your grid will look at runtime.

Resizing

The width of the grid or the selected group can be changed by entering a value (in twips) in the text boxes labeled "Grid Width" and "Group Width".

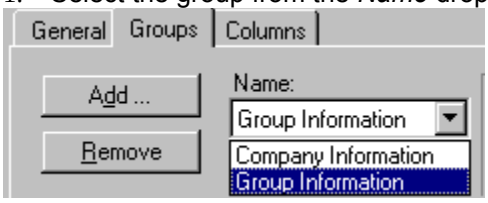
Alternatively, you can resize the width of the grid by dragging the splitter, and you can resize the width of a group by clicking on the right-edge of its header and dragging the group to the desired size.

Adding a group to the Design Grid

1. Click the **Add** button.
2. Specify the name for the group in the "Add Group" dialog.
   The group will be added to the grid.

Removing a group from the Design Grid

1. Select the group from the *Name* drop-down list.



2. Click the **Remove** button.

Working with group properties

There are certain properties that are group-specific. These properties can be easily changed through the Grid Editor.

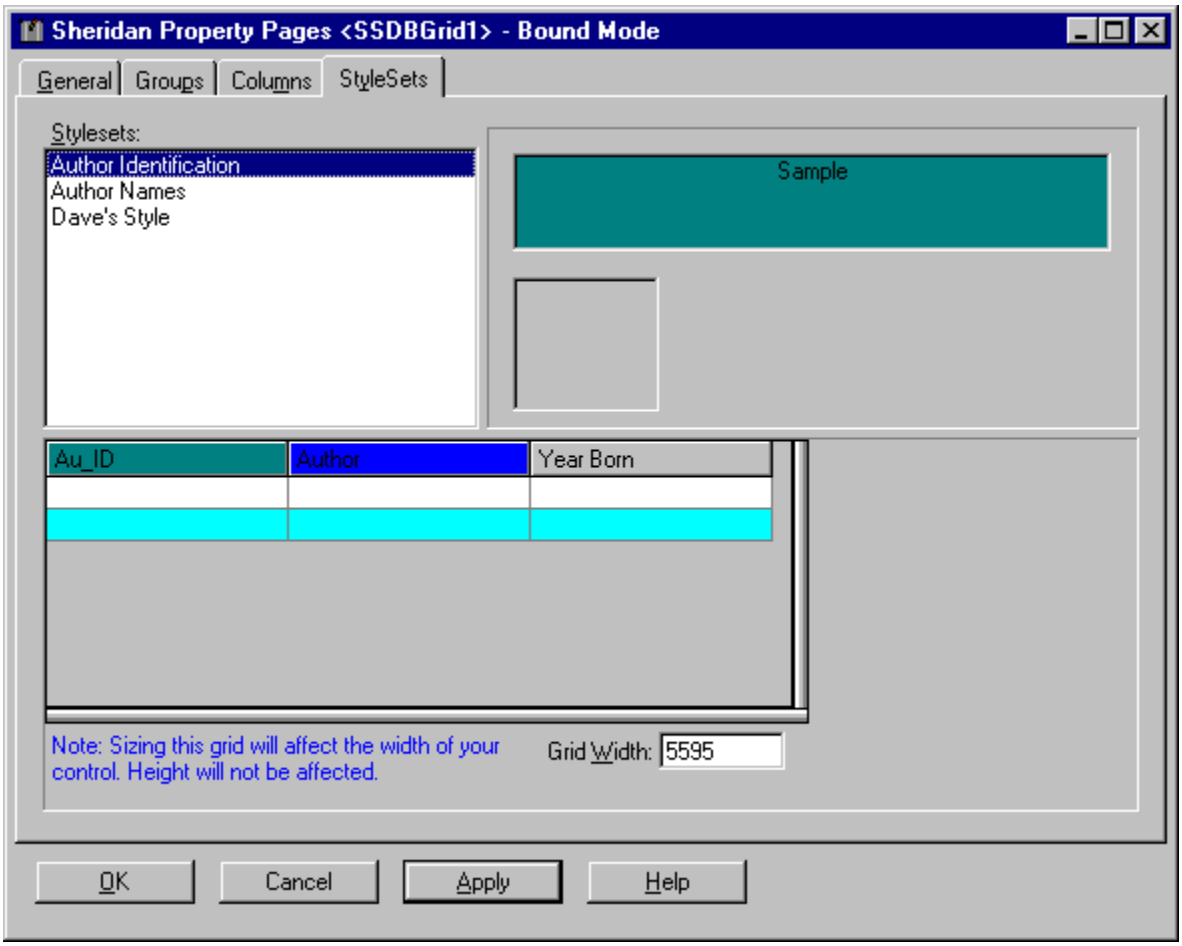
To set group specific properties:

1. Select the group from the *Name* drop-down list.
2. Select the property to modify from the tree and make the desired changes from the options presented on the right.

Grid Editor: StyleSets tab

With the StyleSets Tab, you are able to apply the StyleSets you have created. Select the StyleSet you want to use from the list, and drag it to the part of the Design Grid you want it applied to. For more information on StyleSets, refer to the StyleSet Property.

When you select a StyleSet, a sample of the attributes it has will appear to the right. Similarly, when you apply a StyleSet, you will see it in the Design Grid.



Group Header

Group Object Applies To

Groups collection

Group Property

See Also Applies To

Description

Returns the current group.

Syntax

object.**Group**[= *number*]

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
number	An integer expression specifying the current group.

Group Property Applies To

SSDBGrid

Group Property See Also

Col
Grp
Row

GroupHeadLines Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

GroupHeadLines Property See Also

HeadLines

GroupHeaders Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

GroupHeaders Property See Also

ColumnHeaders

Groups Collection Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Groups Collection See Also

Group object

Groups Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Grp Property

See Also Applies To

Description

Sets or returns the current group.

Syntax

object.**Grp**[= *number*]

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
number	An integer expression specifying the group that the column belongs to.

Remarks

The **Grp** property is useful for returning the group number that the column is a member of. It can also be used to set the group number that the column belongs to.

Grp Property Applies To

Column object

Grp Property See Also

Col
Group
Row

GrpContaining Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

GrpHeadClick Event Applies To

SSDBGrid

GrpHeadClick Event See Also

GrpResize event
HeadClick event

GrpMove Event Applies To

SSDBGrid

GrpMove Event See Also

ColMove event
ColSwap event
GrpSwap event

GrpPosition Method Applies To

SSDBGrid

GrpPosition Method See Also

ColPosition method
Position property

GrpResize Event Applies To

SSDBGrid

GrpResize Event See Also

GrpHeadClick
HeadClick
ResizeWidth

GrpSwap Event Applies To

SSDBGrid

GrpSwap Event See Also

ColMove event
ColSwap event
GrpMove event

Guided Tours

Data Grid control
Sample programs using the Data Grid (Chapter 5)

Exercise 1: Creating a Bound Data Grid
Exercise 2: Creating an Unbound Data Grid
Exercise 3: Creating an AddItem data Grid

Data Combo Control
Sample programs using the Data Combo (Chapter 6)

Exercise 1: Creating an application using the Data Combo
Exercise 2: Customizing the Data Combo

Data DropDown Control
Sample program using the Data DropDown (Chapter 7)

Exercise 1: Creating an application using the Data DropDown

DataOptionSet Control
Sample program using the Data DropDown (Chapter 8)

Exercise 1: Creating an application using the DataOptionSet

Enhanced Data Control Control
Sample programs using the Enhanced Data Control (Chapter 9)

Exercise 1: Creating an application using the EDC
Exercise 2: Using the Find feature of the EDC
Exercise 3: Using Bookmarks in the Enhanced Data Control

Data Command Control
Sample program using the Data Command Button (Chapter 10)

Exercise 1: Creating an application using the Data Command

Biblio File Structure

HasBackColor Property Applies To

Column object

HasBackColor Property See Also

HasForeColor

HasForeColor Property Applies To

Column object

HasForeColor Property See Also

HasBackColor

HasHeadBackColor Property Applies To

Column object
Group object

HasHeadBackColor Property See Also

HasHeadForeColor
HeadBackColor
HeadForeColor

HasHeadForeColor Property Applies To

Column object
Group object

HasHeadForeColor Property See Also

HasHeadForeColor
HeadBackColor

HeadBackColor Property Applies To

Column object
Group object

HeadBackColor Property See Also

HasHeadBackColor
HasHeadForeColor
HeadForeColor

HeadClick Event Applies To

SSDBGrid

HeadClick Event See Also

GrpResize event
HeadClick event

HeadFont Object Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

HeadFont Object See Also

Bold
Font3D
Italic
Name
Size
Strikethrough
Underline

HeadFont3D Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

HeadForeColor Property Applies To

Column object
Group object

HeadForeColor Property See Also

HasHeadBackColor
HasHeadForeColor
HeadBackColor

HeadLines Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

HeadLines Property See Also

GroupHeadLines

HeadStyleSet Property Applies To

Column object
Group object
SSDBCombo
SSDBDropDown
SSDBGrid

HeadStyleSet Property See Also

HeadStyleSet
StyleSet object
StyleSets collection

HeightGap Property Applies To

SSDBOptSet

Included Files

The Setup program will place OCX files in the directories you have specified. Sample applications from the manual are located in the \SAMPLES subdirectory of the Data Widgets home directory.

The following table gives a brief description of the files that may have been installed on your hard disk during the Setup process. Data Widgets selectively installs support files based on the version numbers of files already installed on your system.

Filename(s)	Description
COMDLG16.OCX	16-Bit Common Dialog OCX (used for demo programs)
COMPOBJ.DLL	Support DLL
DAO2516.DLL	Support DLL
MFC40.DLL	Support DLL (Microsoft Foundation Class DLL)
MFCO40.DLL	Support DLL (Microsoft Foundation Class DLL)
MFCANS32.DLL	Support DLL (Microsoft Foundation Class DLL)
MSAJT200.DLL	Support DLL (compatibility layer DLL)
MSJETERR.DLL	Support DLL (compatibility layer DLL)
MSJETINT.DLL	Support DLL (compatibility layer DLL)
MSOUTL16.OCX	16-Bit Outline Control OCX (used for demo programs)
MSVC40.DLL	Support DLL (Microsoft VC DLL)
MSVCRT20.DLL	Support DLL (Microsoft VC DLL)
OC30.DLL	Support DLL (32-Bit data binding DLL)
OLE2.DLL	Support DLL (OLE DLL)
OLE2DISP.DLL	Support DLL (OLE DLL)
OLE2NLS.DLL	Support DLL (OLE DLL)
OLEPRO32.DLL	Support DLL (OLE DLL)
OLE2PROX.DLL	Support DLL (OLE DLL)
OLE2CONV.DLL	Support DLL (OLE DLL)
OLE2.REG	Support file (OLE registration file)
README.TXT	Data Widgets late-breaking information
SCP.DLL	Support DLL
SSCMD16.EXE	16-Bit SSDBCommand custom property pages
SSCMD32.EXE	32-Bit SSDBCommand custom property pages
SSDATA16.OCX	16-Bit OCX containing SSDBData, SSDBOptSet, SSDBCommand
SSDATA32.OCX	32-Bit OCX containing SSDBData, SSDBOptSet, SSDBCommand
SSDATB16.OCX	16-Bit OCX containing SSDBGrid, SSDBDropDown, SSDBCombo
SSDATB32.OCX	32-Bit OCX containing SSDBGrid, SSDBDropDown, SSDBCombo
SSDATWD2.HLP	Data Widgets Online Help
SSDATWDG.LIC	Data Widgets license file
SSDBDATA.BMP	SSDBData button bitmap

SSDODEMO.EXE	DataOptionSet demo
SSDOS16.EXE	16-Bit SSDBOptSet custom property pages
SSDOS32.EXE	32-Bit SSDBOptSet custom property pages
SSEDC16.EXE	16-Bit SSDBData custom property pages
SSEDC32.EXE	32-Bit SSDBData custom property pages
SSGRID16.EXE	16-Bit SSDBGrid Layout Editor
SSGRID32.EXE	32-Bit SSDBGrid Layout Editor
SSPP16.DLL	16-Bit Property Pages DLL
SSPP32.DLL	32-Bit Property Pages DLL
STDOLE.TLB	Support File
STORAGE.DLL	Support DLL (System DLL)
TABCTL16.OCX	16-Bit Tab Control OCX (used for demo programs)
THREED16.OCX	16-Bit 3D Control OCX (used for demo programs)
TYPELIB.DLL	Support DLL (type library DLL)
UNBOUND.MDB	Unbound grid sample database file
UNINSTALL.EXE	Data Widgets 2.0 uninstall program
VAEN2.DLL	Support DLL
VAEN21.OLB	Support DLL
VBAJET.DLL	Support DLL (compatibility layer DLL)
VBEN16.DLL	Support DLL
VB40016.DLL	16-Bit Visual Basic Runtime DLL
VB40032.DLL	32-Bit Visual Basic Runtime DLL
VBDB16.DLL	Support DLL (Visual Basic database DLL)
\SAMPLES	Directory containing projects demonstrated in Chapters 5 - 10

IndexSelected Property Applies To

SSDBOptSet

InitColumnProps Event Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Select a related topic:

Object Concepts
Property Pages
Fonts

Introduction to OCX controls

Related Topics

What is an OCX control?

An OCX control is a specific type of program that makes use of *Object Linking and Embedding* (OLE) to provide functions to other programs. Because it gives programs something they did not originally have, an OCX control is known as an OLE *server*, and the program that uses its services is an OLE *client*. OCX controls can provide a nearly unlimited range of functions to their clients.

How is an OCX control different from a VBX control?

The VBX control specification was designed exclusively for use with Visual Basic. Although some other languages offer limited VBX support, the majority of VBX controls function only in Visual Basic. VBX controls are also limited in other ways. Their 16-bit architecture restricts their ability to use memory and to function in a 32-bit operating system, such as Windows NT.

The difference between OCX and VBX controls may not even be apparent to you if you program exclusively in Visual Basic. You access the properties of an OCX control at design time and through code just as you do the properties of a VBX. The process of including both types of controls in your project and distributing them is very similar. The similarities end when you move outside of the Visual Basic programming environment.

OCX controls are supported by a much wider range of platforms, including other languages, database management systems, and productivity applications. OCX controls can be used as the building blocks in a modular software environment, where a complete project might include your own code, custom controls and commercial applications all working together. OCX controls also have the ability to make full use of the newest 32-bit operating systems, taking advantage of improved memory access, better multi-tasking and increased performance.

When should I use OCX controls?

OCX controls come in two varieties: 16-bit and 32-bit. 16-bit controls offer compatibility with Windows and Windows for Workgroups 3.1 and 3.11. 32-bit controls work with systems running Windows NT and Windows 95. In general, you should use the most advanced version of the control that is available and is supported by your host environment.

If you are using a 32-bit programming system to develop an application that will run exclusively on a 32-bit platform, use the 32-bit OCX. If you are developing an application that must run on a mixed platform, you can use a 16-bit OCX, although you will obtain better performance if you develop separate 16-bit and 32-bit versions of your program, using the appropriate OCX controls. If you are developing exclusively for a 16-bit platform, use the 16-bit OCX.

IsItemInList Method Applies To

SSDBCombo
SSDBGrid

IsItemInList Method See Also

IsTextValid method

IsTextValid Method Applies To

SSDBCombo

IsTextValid Method See Also

IsItemInList

Italic Property Applies To

Font object
Headfont object

Keyboard Interface

The following describes the keyboard interface for each of the Data Widgets controls that support keyboard use.

SSDBGrid

Press	To	Comments
F4	Toggles dropdown	Only works in cells with a dropdown.
ALT + UP ARROW	Toggles dropdown	Only works in cells with a dropdown.
ALT + DOWN ARROW	Toggles dropdown	Only works in cells with a dropdown.
UP ARROW	Moves up a row in the grid	
DOWN ARROW	Moves down a row in the grid	
PAGE UP	Moves up a page in the grid	
PAGE DOWN	Moves down a page in the grid	
LEFT ARROW	Moves one cell to the left. When in edit mode, moves one character to the left.	
RIGHT ARROW	Moves one cell to the right When in edit mode, moves one character to the right.	
HOME	When in edit mode, moves to the beginning of the cell	
END	When in edit mode, moves to the end of the cell	
ESC	Restores the cell value to what it was prior to entering the cell.	
TAB	Moves one cell forward	
SHIFT + TAB	Moves one cell backward	
CTRL + X	Deletes the selected row	In the case of multiple rows being selected, they will all be deleted. **AllowDelete** must be set to **True.**
DEL	Deletes the selected row	In the case of multiple rows being selected, they will all be deleted.

		AllowDelete must be set to **True.**

SSDBCombo

Press	To	Comments
F4	Toggles the Data Combo's dropdown.	If the dropdown is open, it will be closed. If it is closed, it will be opened.
ALT + UP ARROW	Toggles the Data Combo's dropdown.	If the dropdown is open, it will be closed. If it is closed, it will be opened.
ALT + DOWN ARROW	Toggles the Data Combo's dropdown.	If the dropdown is open, it will be closed. If it is closed, it will be opened.
UP ARROW	Moves up a row	Only works in the dropdown portion.
DOWN ARROW	Moves down a row	Only works in the dropdown portion.
PAGE UP	Moves up a page	Only works in the dropdown portion.
PAGE DOWN	Moves down a page	Only works in the dropdown portion.
LEFT ARROW	Moves one character to the left	Works in the edit portion only.
RIGHT ARROW	Moves one character to the right	Works in the edit portion only.
HOME	Moves to the beginning of the cell	Works in the edit portion only.
END	Moves to the end of the cell	Works in the edit portion only.
ESC	When dropped down, closes the dropdown and restores the value to what it was before dropping down.	
	When not dropped down, restores the text to the previous database value.	
ENTER	When dropped down, selects the current row and closes the dropdown.	Works only on the dropdown portion.

SSDBDropDown

Press	To	Comments
F4	Toggles the dropdown.	Causes the dropdown to close up.
ALT + UP ARROW	Toggles the dropdown.	Causes the dropdown to

		close up.
ALT + DOWN ARROW	Toggles the dropdown.	Causes the dropdown to close up.
UP ARROW	Moves up a row	
DOWN ARROW	Moves down a row	
PAGE UP	Moves up a page	
PAGE DOWN	Moves down a page	
ESC	Closes the dropdown and restores the value to what it was before dropping down.	
ENTER	Selects the current row and closes up the dropdown.	

SSDBOptSet

Press	To	Comments
UP ARROW	Moves up a button	
DOWN ARROW	Moves down a button	
LEFT ARROW	Moves up a button	
RIGHT ARROW	Moves down a button	
HOME	Moves to the first button in the set	
END	Moves to the last button in the set	

LeftCol Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

LeftCol Property See Also

LeftGrp

LeftGrp Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

LeftGrp Property See Also

LeftCol

Level 0 (Multi-Level Row)

Level 1 (Multi-Level Row)

Level Property Applies To

Column object

LevelCount Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

List Property Applies To

Column object

ListAutoPosition Property Applies To

SSDBCombo
SSDBDropDown

ListAutoValidate Property Applies To

SSDBCombo
SSDBDropDown

ListWidth Property Applies To

SSDBCombo
SSDBDropDown

ListWidth Property See Also

ListWidthAutoSize

ListWidthAutoSize Property Applies To

SSDBCombo

Locked Property Applies To

Column object

MaintainBtnHeight Property Applies To

SSDBOptSet

MaxDropDownItems Property Applies To

SSDBCombo
SSDBDropDown

MaxDropDownItems Property See Also

MinDropDownItems

MinColWidth Property Applies To

SSDBOptSet

MinDropDownItems Property Applies To

SSDBCombo
SSDBDropDown

MinDropDownItems Property See Also

MaxDropDownItems

MinHeight Property Applies To

SSDBOptSet

MouseIcon Property Applies To

SSDBCombo
SSDBCommand
SSDBData
SSDBOptSet

MouseIcon Property See Also

MousePointer

MousePointer Property Applies To

SSDBCombo
SSDBCommand
SSDBData
SSDBOptSet

MousePointer Property See Also

MouseIcon

MoveFirst Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

MoveFirst Method See Also

MoveLast
MoveNext
MovePrevious
MoveRecords

MoveLast Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

MoveLast Method See Also

MoveFirst
MoveNext
MovePrevious
MoveRecords

MoveNext Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

MoveNext Method See Also

MoveFirst
MoveLast
MovePrevious
MoveRecords

MovePrevious Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

MovePrevious Method See Also

MoveFirst
MoveLast
MoveNext
MoveRecords

MoveRecords Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

MoveRecords Method See Also

MoveFirst
MoveLast
MoveNext
MovePrevious

Name Property Applies To

Font object
Headfont object
StyleSet object

NumberFormat Property Applies To

Column object

NumberFormat Property See Also

Visual Basic's **Format** Function

NumberOfButtons Property Applies To

SSDBOptSet

NumberOfButtons Property See Also

IndexSelected

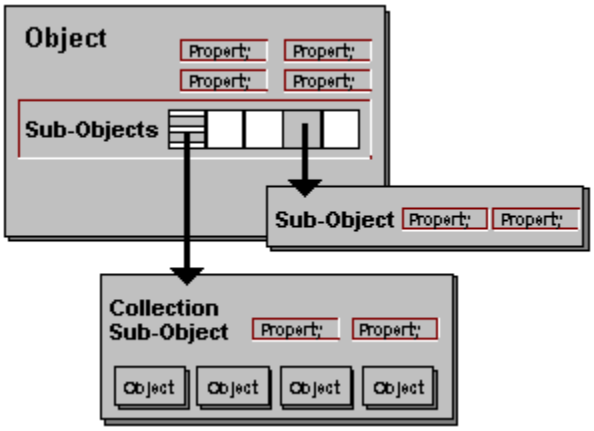
Object Concepts

This section will be of special interest to programmers who have worked with earlier versions of our custom controls. It highlights the major differences between the older controls you may be familiar with and the newer controls you now have.

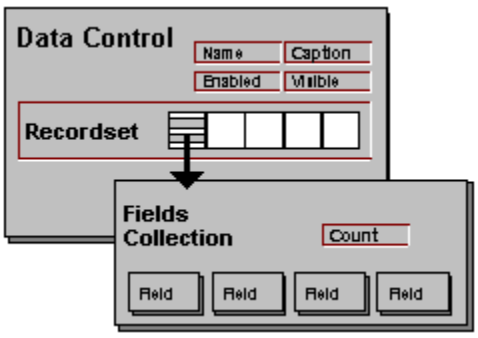
Object-oriented programming offers you greater power than before, with less work on your part. However, because this is a new technology, there are some new concepts with which you should be familiar. This section provides a brief introduction to some of the new concepts you will encounter while using Sheridan custom controls.

Sub-objects and Collections

Data Widgets provides an object-oriented approach to programming through the use of *sub-objects* and *collections*. An *object* refers to a single unit or entity within your application which contains both code and data. Objects can contain other objects, which have properties and methods of their own that can be examined and changed. Objects may also contain *collection* objects. A collection is a special type of object that contains sub-objects that are all of the same type, or *class.*



You are probably familiar with the concept of sub-objects if you have used the Visual Basic data control. The **Recordset** object is a sub-object of the Visual Basic Data Control. The **Recordset** contains a collection sub-object called the **Fields** object, which contains information that relates to all the fields in the **Recordset** collectively. The **Fields** collection also contains the **Field** objects themselves, which store data and also information pertaining to that data.



Objects within collections often have this type of "paired" arrangement; a single collection object (**Fields**) which describes and contains the collection as a whole, and multiple member objects (**Field**) which make up the collection. In addition, there is usually a corresponding property of the same name as the object

that returns information about the object.
Collections have replaced property arrays as the preferred method for accessing sets of controls at runtime. This means you no longer have to specify an array for each property you wish to access, and there are fewer special property names. For example, previously to set the caption text of the fifth tab in a DataOptionSet control, you would have used the following code: SSDBOptSet1.Caption(4) = "5th Button"

Now, you would use the standard **Caption** property, specifying instead the object in the collection to which it will apply: SSDBOptSet1.Buttons(4).Caption = "5th Button"

This control's object type is:

SSDBCombo

This control's object type is:

SSDBCommand

This control's object type is:

SSDBDropDown

This control's object type is:

SSDBGrid

This control's object type is:

SSDBOptSet

This control's object type is:

SSDBData

Odd Row (Row 1)

Optimizing Data Widgets

Improving Load Time

By default, the Data Grid, Data Combo, and Data DropDown custom controls each go to the last record in a record set to determine the exact number of rows. The controls do this to give an accurate number of rows in the **Rows** property. However, with large databases, this could cause a decrease in performance.

To turn this option off, set the **UseExactRowCount** property to **False**. This will cause the control to estimate the number of rows in the record set. If this property is set to **False**, *do not rely on the **Rows** property for an accurate number of rows.* If you do a **MoveLast** on the data control's record set, the **Rows** property will be accurate.

Optimizing the Data Combo and Data DropDown

The Data Combo and Data DropDown can be optimized when performing certain functions. There are two functions that the controls perform automatically which can be overridden.

Auto List Validation

The Data Combo and Data DropDown automatically perform validation of the value in the edit portion of the Data Combo or the cell of a Data Grid against the values in the list portion to find a match. in some circumstances, this validation can be very slow, since the control must sequentially search the entire database. With large databases, this operation can be quite slow. To turn this feature off and perform you own validation of the field, set the **ListAutoValidate** property to **False**. This will cause the control to skip the validation process and trigger the **ValidateList** event.

Auto Positioning

Another specific way of optimizing the performance of the Data Combo or Data DropDown controls is to set the **ListAutoPosition** property to **False**. This turns off the automatic positioning of the list based on the contents of the edit portion of the Data Combo or cell of the Data Grid. Instead, the **PositionList** event will be triggered.

Similar to the validation of data against the list, the positioning requires the control to search the list sequentially which can cause a performance penalty with large databases.

OptionValue Property Applies To

Button object
SSDBOptSet

OptionValue Property See Also

DataField
DataSource

Orientation Property Applies To

SSDBData

PageValue Property Applies To

SSDBCommand
SSDBData

Picture Property Applies To

Button object
SSDBCommand
SSDBOptSet

Picture Property See Also

AutoSize
PictureMetaHeight
PictureMetaWidth
PictureAlignment

PictureAlignment Property Applies To

SSDBCommand
SSDBOptSet

PictureAlignment Property See Also

CaptionAlignment

PictureButton Property Applies To

SSDBGrid

PictureButton Property See Also

PictureComboButton
PictureRecordSelectors

PictureButtons Property Applies To

SSDBData

PictureButtons Property See Also

PictureCaption
PictureCaptionAlignment

PictureCaption Property Applies To

SSDBData

PictureCaption Property See Also

PictureButtons
PictureCaptionAlignment

PictureCaptionAlignment Property Applies To

SSDBData

PictureCaptionAlignment Property See Also

PictureButtons
PictureCaption

PictureComboButton Property Applies To

SSDBGrid

PictureComboButton Property See Also

PictureButton
PictureRecordSelectors

PictureDropDown Property Applies To

SSDBCombo

PictureMetaHeight Property Applies To

Button object
SSDBCommand
SSDBOptSet

PictureMetaHeight Property See Also

Picture
PictureMetaWidth

PictureMetaWidth Property Applies To

Button object
SSDBCommand
SSDBOptSet

PictureMetaWidth Property See Also

Picture
PictureMetaHeight

PictureRecordSelectors Property Applies To

SSDBGrid

PictureRecordSelectors Property See Also

PictureButton
PictureComboButton

Position Property Applies To

Column object
Group object

Position Property See Also

ColPosition method
GrpPosition method

PositionList Event Applies To

SSDBCombo
SSDBDropDown

PositionList Event See Also

ListAutoPosition

Property Pages

Sheridan Software custom controls support a feature known as property pages. Property pages provide an interface through which you can view and modify the properties of your custom control objects. The purpose of property pages is twofold. First, property pages allow you to set properties at design time that would not otherwise be available - the so-called "runtime" properties. Second, property pages allow you to modify your control in a host environment that does not provide a property sheet.

The Property Pages Interface

The property pages provide access to a different aspect of a control's behavior. What appears in a given dialog will depend on the features that the control supports. There will always be at least one tab called 'Properties' which lists all the properties of the control. Other tabs may support added functions or utilities.

Properties are listed in a hierarchical menu structure similar to the tree view of the Windows File Manager. This structure makes it easy for you to access the properties of sub-objects and collections. As you choose a property name from the tree on the left, the valid settings for that property appear on the right, enabling you to examine or modify them.

Accessing Property Pages

The method you use to access the property pages of your control depends on two things; the version of the control you are using, and the host environment in which you are using the control.

Many host environments support the use of the right mouse button to pop up a context-specific menu. In these environments, you simply click on your control with the right mouse button, and choose 'Property Pages' or 'Properties' from the pop-up menu.

If this behavior is not supported, use the property sheet of your design environment. You will see a property labeled '(Custom)' in the property sheet. By double-clicking this property or choosing the ellipsis (...) button, you can invoke the property pages for the selected control.

If neither of these methods are supported, you will need to consult the documentation of your host environment for information on how to change the properties of objects. You may need to choose a special menu option, or perform a shifted mouse-click or double-click on the control. Try searching your environment's online help file for references to *objects, embedded objects, object properties, object settings, OLE linking, OLE servers,* or *properties.*

Note For the SSDBCombo, SSDBDropDown, and SSDBGrid controls, property pages are replaced by the Grid Editor which performs all functions of a property page.

Rebind Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Record Selectors

RecordSelectors Property Applies To

SSDBGrid

Redraw Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Remove Method Applies To

Bookmarks collection
Buttons collection
Columns collection
Groups collection
SelBookmarks collection
StyleSets collection

Remove Method See Also

Add method
Count property
RemoveAll method

RemoveAll Method (AddItem Mode) Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

RemoveAll Method (Collections) Applies To

Bookmarks collection
Buttons collection
Columns collection
Groups collection
SelBookmarks collection
StyleSets collection

RemoveAll Method (Collections) See Also

Add method
Remove method

RemoveAll Method (Column Object) Applies To

Column object

RemoveItem Method (AddItem Mode) Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

RemoveItem Method (Column Object) Applies To

Column object

Reset Method

Applies To

Description

Destroys the associated layout for a control.

Syntax

object.**Reset**

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.

Remarks

The reset method is useful for when the programmer changes the DataSource and needs to create a new layout.

Example

The following example resets the layout, changes the data mode, and creates a new layout:

```
SSDBGrid1.Reset
SSDBGrid1.DataMode = 2
SSDBGrid1.Cols = 2
SSDBGrid1.Columns(0).Caption = "Name"
SSDBGrid1.Columns(1).Caption = "Social Security Number"
SSDBGrid1.Refresh                                    ‘ Needed to display new layout
```

Reset Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

ResizeHeight Property Applies To

SSDBGrid

ResizeHeight See Also

ResizeWidth

ResizeWidth Property Applies To

SSDBGrid

ResizeWidth Property See Also

ResizeHeight

RotateText Property Applies To

SSDBData

RotateText Property See Also

Caption
Orientation

RoundedCorners Property Applies To

SSDBCommand
SSDBData

Row Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Row Property See Also

Col
Grp
VisibleRows

RowBookmark Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

RowChanged Property Applies To

SSDBGrid

RowColChange Event Applies To

SSDBGrid

RowColChange Event See Also

RowLoaded

RowContaining Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

RowContaining Method See Also

ColContaining

RowHeight Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

RowHeight Property See Also

DefColWidth

RowLoaded Event Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

RowLoaded Event See Also

RowColChange

RowNavigation Property Applies To

SSDBGrid

RowNavigation Property See Also

CellNavigation

RowOffset Property Applies To

Button object
SSDBOptSet

RowOffset Property See Also

ColOffSet

RowResize Event Applies To

SSDBGrid

RowSelectionStyle Property

See Also Applies To

Description

Determines how a row will appear when selected..

Syntax

object.**RowNavigation**[= *number*]

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
number	An integer expression specifying how a row appears when selected..

Settings

Setting	Description
0	Invert colors.
1	(Default) ListBox style (using System colors for highlight).
2	3D appearance.

RowSelectionStyle Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

RowSelectionStyle Property See Also

ActiveRowStyleSet

RowTop Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Rows Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Rows Property See Also

Col
Cols
Row

This control is located in:

SSDATA16.OCX, SSDATA32.OCX

This control is located in:

SSDATB16.OCX, SSDATB32.OCX

SavedBookmark Property Applies To

SSDBCommand

Scroll Event Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Scroll Event See Also

Scrollbars
Scroll method

Scroll Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Scroll Method See Also

Scrollbars
Scroll event

Scrollbars Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Scrollbars Property See Also

Scroll event
Scroll method

SelBookmarks Collection Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

SelBookmarks Method

Applies To

Description

Returns a Bookmark object at the specified index.

Syntax

object.**SelBookmarks***(* [*Index* **As Variant**]*)*

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
Index	A variant specifying the bookmark number.

Remarks

When no index is specified the SelBookmarks collection object is returned.

SelBookmarks Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

SelChange Event

Applies To

Occurs when the current range changes to a different cell or range of cells.

Syntax

Sub control_SelChange (*numselected* **As Long,** *SelType* **As Long**, *cancel* **As Integer**)

The event parameters are:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
numselected	*Indicates the number of rows or columns selected.*
seltype	*Indicates the type of selection (0=Group, 1=Column, 2=Row).*
cancel	Determines whether the selection reverts to its position before the event occurred.

Remarks

Occurs when a cell other than the current is clicked as well as when a user drags to select a new range of cells.

Setting cancel to True causes the selection to revert to the cell or range active before the event occurred.

SelChange Event Applies To

SSDBGrid

SelectByCell Property Applies To

SSDBGrid

SelectByCell Property See Also

SelectionTypeCol
SelectionTypeRow

Selected Property Applies To

Column object
Group object

SelectionTypeCol Property Applies To

SSDBGrid

SelectionTypeCol Property See Also

SelectionTypeRow

SelectionTypeRow Property Applies To

SSDBGrid

SelectionTypeRow Property See Also

SelectionTypeCol

ShowAddButton Property Applies To

SSDBData

ShowAddButton Property See Also

ShowBookmarkButtons
ShowCancelButton
ShowDeleteButton
ShowFindButtons
ShowFirstLastButtons
ShowPageButtons
ShowPrevNextButtons
ShowUpdateButton

ShowBookmarkButtons Property Applies To

SSDBData

ShowBookmarkButtons Property See Also

ShowAddButton
ShowCancelButton
ShowDeleteButton
ShowFindButtons
ShowFirstLastButtons
ShowPageButtons
ShowPrevNextButtons
ShowUpdateButton

ShowBookmarkDropDown Event Applies To

SSDBData

ShowBookmarkDropDown Event See Also

DroppedDown
CloseBookmarkDropDown Method

ShowCancelButton

See Also Applies To

Description

Determines whether the Cancel button is displayed on the control.

Syntax

object.**ShowCancelButton**[= *boolean*]

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
boolean	A boolean expression specifying the display state of the Cancel button on the control.

Settings

Setting	Description
True	(Default) The Cancel button will be displayed.
False	The Cancel button will not be displayed.

Remarks

The Cancel (Cancel Add) button cancels the adding of a new record to the database.

ShowCancelButton Applies To

SSDBData

ShowCancelButton See Also

ShowAddButton
ShowBookmarkButtons
ShowDeleteButton
ShowFindButtons
ShowFirstLastButtons
ShowPageButtons
ShowPrevNextButtons
ShowUpdateButton

ShowDeleteButton Property Applies To

SSDBData

ShowDeleteButton Property See Also

ShowAddButton
ShowBookmarkButtons
ShowCancelButton
ShowFindButtons
ShowFirstLastButtons
ShowPageButtons
ShowPrevNextButtons
ShowUpdateButton

ShowFindButtons Property Applies To

SSDBData

ShowFindButtons Property See Also

ShowAddButton
ShowBookmarkButtons
ShowCancelButton
ShowDeleteButton
ShowFirstLastButtons
ShowPageButtons
ShowPrevNextButtons
ShowUpdateButton

ShowFindDialog Event Applies To

SSDBData

ShowFindDialog Event See Also

FindDialog
CloseFindDialog event

ShowFirstLastButtons Property Applies To

SSDBData

ShowFirstLastButtons Property See Also

ShowAddButton
ShowBookmarkButtons
ShowCancelButton
ShowDeleteButton
ShowFindButtons
ShowPageButtons
ShowPrevNextButtons
ShowUpdateButton

ShowPageButtons Property Applies To

SSDBData

ShowPageButtons Property See Also

ShowAddButton
ShowBookmarkButtons
ShowCancelButton
ShowDeleteButton
ShowFindButtons
ShowFirstLastButtons
ShowPrevNextButtons
ShowUpdateButton

ShowPrevNextButtons Property Applies To

SSDBData

ShowPrevNextButtons Property See Also

ShowAddButton
ShowBookmarkButtons
ShowCancelButton
ShowDeleteButton
ShowFindButtons
ShowFirstLastButtons
ShowPageButtons
ShowUpdateButton

ShowUpdateButton Property Applies To

SSDBData

ShowUpdateButton Property See Also

ShowAddButton
ShowBookmarkButtons
ShowCancelButton
ShowDeleteButton
ShowFindButtons
ShowFirstLastButtons
ShowPageButtons
ShowPrevNextButtons

Size Property Applies To

Font object
HeadFont object

Speed Buttons

Speed buttons allow for a user to click on a button and hold it to repeat a function.

This function is controlled by the **DelayInitial** and **DelaySubsequent** properties. **DelayInitial** determines the amount of time before a speed button begins to repeat when the mouse button is held down. **DelaySubsequent** determines the amount of time between subsequent clicks are repeated when the mouse button is held down on a repeatable button.

SplitterMove Event Applies To

SSDBGrid

SplitterMove Event See Also

SplitterPos
SplitterVisible

SplitterPos Property Applies To

SSDBGrid

SplitterPos Property See Also

SplitterVisible
SplitterMove Event

SplitterVisible Property Applies To

SSDBGrid

SplitterVisible Property See Also

SplitterPos
SplitterMove Event

Standard Property - Depending on your host environment, this property may be referred to by a different name or may not apply to this control. Refer to your host environment's documentation or help file for further information regarding this property.

Strikethrough Property Applies To

Font object
HeadFont Object

String Property Applies To

Bookmark object

String Property See Also

Value

Style Property Applies To

Column object

StyleSet Object Applies To

StyleSets collection

StyleSet Object See Also

HeadStyleSet
StyleSet
StyleSets collection.

StyleSet Property Applies To

ActiveCell object
Column object
Group object
SSDBCombo
SSDBDropDown
SSDBGrid

StyleSet Property See Also

HeadStyleSet
StyleSet object
StyleSets collection

StyleSets Collection Applies To

ActiveCell object
Column object
Group object
SSDBCombo
SSDBDropDown
SSDBGrid

StyleSets Collection See Also

HeadStyleSet
StyleSet
StyleSet object

StyleSets Method

Applies To

Description

Returns a StyleSet object at the specified index.

Syntax

object.**StyleSets***(*** [*Index* **As Variant**]***)***

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
Index	A variant specifying the StyleSet number.

Remarks

When no index is specified the StyleSets collection object is returned.

StyleSets Method Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

TagVariant Property Applies To

Column object
Group object
SSDBCombo
SSDBCommand
SSDBDropDown
SSDBGrid
SSDBOptSet

Technical Specifications

You must have the following to utilize Data Widgets:

Microsoft Visual Basic version 4.x or a development tool that supports OLE Custom Controls (.OCX files).

A hard disk with at least 5 megabytes of available space for a full installation.

For the 32-bit version of Data Widgets, you must have Windows 95 or later, or Windows NT 3.51 or later.

For the 16-bit version of Data Widgets, you must have Windows version 3.1 or later, running in enhanced mode.

Included Files

Distributing Your Application

Error Messages

Technical Support

World Wide Web

The Sheridan Software World Wide Web site provides the latest patches and product information, as well as information for the Visual Basic developer.

http://www.shersoft.com

Internet Email

Submit your questions to our technical support staff via electronic mail. Be sure to include detailed information on your problem, the Sheridan product and product version you are using, as well as information on your host environment such as the machine type, RAM, video card, and operating system.

support@shersoft.com

CompuServe

You can obtain technical support on CompuServe by contacting the SYSOP at the SHERIDAN section of the COMPA forum. You can type GO SHERIDAN at any CompuServe prompt.

Bulletin Board Service (BBS)

For free upgrades to Sheridan products, connect to the Sheridan BBS at (516) 753-5452. Have your modem set to 8N1.

Fax

To fax questions or comments regarding any Sheridan product, dial (516) 753-3661.

Telephone Support

For free technical support for this or any other Sheridan product, contact Sheridan Software systems at (516) 753-0985. You can either speak to a live technical support representative or get answers using the Automated Fax Service. Sheridan's support hours are 9AM to 5PM (EST), Monday through Friday.

Text Property Applies To

ActiveCell object
Column object
SSDBCombo

TextError Event Applies To

SSDBCombo
SSDBDropDown

TextError Event See Also

ListAutoValidate

TextFormat Property Applies To

SSDBCombo

UnboundAddData Event Applies To

SSDBGrid

UnboundAddData Event See Also

AllowAddNew property
UnboundPositionData event
UnboundReadData event
UnboundWriteData event
ssRowBuffer Object

UnboundDeleteRow Event Applies To

SSDBGrid

UnboundDeleteRow Event See Also

AllowDelete property
UnboundAddData event
UnboundPositionData event
UnboundReadData event
UnboundWriteData event
ssRowBuffer Object

UnboundPositionData Event Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

UnboundPositionData Event See Also

UnboundAddData event
UnboundDeleteRow event
UnboundReadData event
UnboundWriteData event
ssRowBuffer Object

UnboundReadData Event Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

UnboundReadData Event See Also

UnboundAddData event
UnboundDeleteRow event
UnboundPositionData event
UnboundWriteData event
ssRowBuffer Object

UnboundWriteData Event Applies To

SSDBGrid

UnboundWriteData Event See Also

UnboundAddData event
UnboundDeleteRow event
UnboundPositionData event
UnboundReadData event
UnboundWriteData event
ssRowBuffer Object

Underline Property Applies To

Font object
Headfont object

Underline Property See Also

Font object
Headfont object

Updating Rows from a Modal Form

There is a caveat when using a bound control to update a row or rows in a record set of a data control on a modal form. If a row is updated with an invalid field, such as a null key field, Visual Basic does not display an error until the modal form is hidden or unloaded. To overcome this Visual Basic limitation, include the following code in response to the **Error** event of the Visual Basic data control:

```
Sub Data1_Error (DataErr As Integer, Response As Integer)
    On Error Resume Next
        If DataErr Then
Beep
MsgBox Error (DataErr)
DataErr = 0
Response = 0
        End If
End Sub
```

Note This is applicable to any bound control including the standard Visual Basic controls.

UseExactRowCount Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

Using Data Widgets

How Data Widgets Are Supplied

Data Widgets ships a total of four OCX files, grouped in pairs by 16-bit and 32-bit controls.

Filename	Controls Contained
SSDATA16.OCX	16-bit version of the Enhanced Data Control, DataOptionSet, and Data Command Button controls.
SSDATB16.OCX	16-bit version of the Data Grid, Data DropDown, and Data Combo controls.
SSDATA32.OCX	32-bit version of the Enhanced Data Control, DataOptionSet, and Data Command Button controls.
SSDATB32.OCX	32-bit version of the Data Grid, Data DropDown, and Data Combo controls.

Including Data Widgets in Your Project

Custom controls are generally installed on a project-need basis. Once you have included a custom control in a project and saved that project, the control will be available whenever you subsequently open the project.

The method you use to add a Data Widgets control to your project varies depending on which programming environment you are using. Data Widgets comes in two varieties:

32-bit OCX controls that are compatible with any programming system supporting OLE custom controls, supports advanced data binding, and runs in a 32-bit environment such as Windows NT or Windows 95.

16-bit OCX controls that are compatible with any programming system supporting OLE custom controls and advanced data binding.

Visual Basic 4.0

The use of OLE Custom Controls are new to Version 4.0 of Visual Basic, replacing the previously used VBX format. To use the Data Widgets controls in Visual Basic 4.0:

1. Open the project you want to add the Data Widgets control to.
2. Select **Custom Controls** from the *Tools* menu.
3. Select the appropriate Data Widgets control from the list of Available Controls.
   All of the Data Widget controls are prefaced by the name Sheridan (i.e., Sheridan Data Command Control). A checked box next to the control indicates that it has been selected.
4. Click the **OK** button.
   The control(s) you have selected are now added to your project.

Other Languages

Data Widgets OCX controls are supported by a variety of host environments. To use Data Widgets in programming environments other that Visual Basic 4.0, consult your development tool's documentation for information on how to use *OLE Custom Controls* or *OCX Controls*.

Once the control is loaded, it should appear as an extension of your environment. Use the control's Property Pages or the environment's property sheet (if available) to set up the control.

For more information on compatibility between different versions of the controls across different host environments, see Introduction to OCX Controls.

Using a Data DropDown in a Data Grid Column

Another powerful feature of the Data Grid is the ability to link a Data DropDown control to a column in the Data Grid. Similar to the cell button feature, this feature allows the user to click a button in the cell to drop down a list of choices.

The Data DropDown control can be bound to another record set in another data control. For instance, if one of the columns in the Data Grid contains a State Code, you can link in a Data DropDown and bind it to a data control with a list of State Codes and descriptions. When a button is clicked, a list of states would drop down for the user to choose from.

This is done by setting the **DropDownHwnd** property to the window handle of a Data DropDown control that is on your form. For more information on how to do this, refer to the Data DropDown.

Using the Cell Button Feature of the Data Grid

Each column in the Data Grid allows you to include on the right of each cell a button for you to perform additional processing when pressed. To activate the cell button feature, set the property **Style** to **Edit Button** for the corresponding column. Whenever the button is clicked, the **BtnClick** event will be triggered, allowing you to perform any function you wish, such as displaying a dialog with a larger text box for memo-type fields.

Using the Data Grid as a List Box

By default, the Data Grid does not look nor act much like a standard Windows list box. However, by setting just a few properties, the Data Grid can look and behave just like one. It can be used as a bound list box or an unbound virtual list.

To make the Data Grid work like a listbox, set the **AllowAddNew**, **AllowDelete**, **AllowDragDrop**, **AllowUpdate**, and **RecordSelectors** properties all to **False**. Set **SelectByCell** to **True, SelectionTypeRow** to either **Single** or **MultiSelect**, and set **SelectionTypeCol** to **None**. The Data Grid can now be used as a multi-column list box with optional headings. You can modify other properties as needed to customize the grid to your liking.

ValidateList Event Applies To

SSDBCombo
SSDBDropDown

Value Property (Bookmark) Applies To

Bookmark object

Value Property (Bookmark) See Also

String

Value Property (Button Object)

Applies To

Description

Sets or returns the current state (checked / not checked) of the button object.

Syntax

object.**Value**[= *boolean*]

Part	Description
object	An object expression that evaluates to an object or a control in the Applies To list.
value	A boolean expression specifying whether the button is selected or not.

Settings

Setting	Description
True	(Default) Button object is selected.
False	Button object is not selected.

Value Property (Button Object) Applies To

Button object

Value Property Applies To

ActiveCell Object
Column Object
ssRowBuffer Object

Value Property See Also

Text

Help File Version

Version 2.00.05 01/24/96

VertScrollBar Property Applies To

Column object

VisibleCols Property Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

VisibleCols See Also

Col
Cols

VisibleGrps Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

VisibleRows Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

What are Bookmarks?

Bookmarks are a powerful feature that allow you to "flag" a record you want to remember. The EDC allows you to Add, Store, and Delete bookmarks without the need for coding. To access a stored bookmark, the user need only click on the  button, where a dropdown list of stored bookmarks will appear for selection.

What is Data Widgets?

Data Widgets is a set of custom controls that allow you to design front-ends for database applications with all the simplicity and power you have come to expect from your host development application.

Designed with ease of use in mind, Data Widgets virtually eliminates the need for time-consuming coding when developing applications involving database operations. What used to take hours of development can now take minutes. All you need to do is drop a control on a form, set a few properties, and Data Widgets does the rest!

Data Widgets includes six bound custom controls, each for specific data-manipulation functions, provided in both 16-bit and 32-bit OLE Custom Control (OCX) format.

What's New?

Perhaps the most significant change in Data Widgets 2.0 is the transition from VBX to OCX format for custom controls. The OCX format utilizes Microsoft's OLE automation specifications. Controls such as Data Widgets can now be used on a range of development environments, whereas in the past, Data Widgets was limited to environments that supported the VBX format.

Version 2.0 of Data Widgets introduces the use of objects and collections. Objects allow you to manipulate the custom controls much more easily and with more power than in the past. For example, you can easily set a property specific to an object (such as an individual column) by accessing the object directly. This means that you can customize Data Widgets to suit your individual needs. Collections are simply organized groupings of objects.

Because of the introduction of objects and collections, code that worked in Version 1.0 will need to be modified to take advantage of the new structure. Additionally, many of the properties that existed in 1.0 have either been altered to conform with this new structure, or eliminated completely.

WhereIs Applies To

SSDBData
SSDBOptSet
SSDBGrid

WhereIs Method See Also

ButtonFromCaption
ButtonFromPos

WidthGap Property Applies To

SSDBOptSet

WidthGap Property See Also

ColOffSet
RowOffSet

WordWrap Property Applies To

SSDBCommand
SSDBOptSet

ssRowBuffer Object Applies To

SSDBCombo
SSDBDropDown
SSDBGrid

ssRowBuffer Object See Also

UnboundAddData event
UnboundDeleteRow event
UnboundReadData event
UnboundWriteData event