

GridEX Control

[See Also](#)

[Properties](#)

[Methods](#)

[Events](#)

Provides an intuitive user interface for data manipulation.

Syntax

GridEX

Remarks

The Janus **GridEX** control allows a programmer to give their database applications a more intuitive user interface. With a **GridEX** control you can sort and group records, also you can present records in a **Recordset** as lines in a grid or as cards.

Each cell of a **GridEX** control can hold text values and/or icons. The cell is displayed depending on its column settings. A column can be displayed as text, icon, icon and text, and as a check box. In addition, a cell could be edited as a text box, check box or drop-down list.

With the **GridEX** control, a user can drag a column header and drop it into a new position or, in the group by box for grouping. When a user updates a record, the GridEX control finds the appropriate position of the record based on the sort and/or group settings and changes in order to remain valid.

You can use the **GridEX** control as a standard grid, as a **ListView** in report mode, or as a cardholder, where each card represents a record in a **Recordset**. You can also use the **GridEX** control in unbound mode to presents data in any format.

AboutBox Method

[See Also](#)

[Example](#)

[Applies To](#)

Displays the About box for the control.

Syntax

object.**AboutBox**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This is the same as clicking About in the Properties window.

ADORecordset Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets an **ADOR.Recordset** object defined by **GridEX** control's properties or by an existing **ADOR.Recordset** object.

Syntax

Set *object*.**ADORecordset** [= *value*]

The **ADORecordset** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	An object variable containing an ADOR.Recordset or an ADODB.Recordset object

Remarks

The **GridEX** control is automatically initialized when the control first appears. If the **DataMode** property is set to **jpgexADO** and the **DatabaseName**, **LockType**, **Options**, **RecordSource** and **RecordsetType** properties are valid, or if you set these properties at run time and use the **Rebind** method, the control attempts to create a new **ADOR.Recordset** object based on those properties. A clone of this **ADOR.Recordset** is accessible through the **GridEX** control's **ADORecordset** property. If, however, one or more of these properties is set incorrectly at design time, an error event is sent to the application when the control attempts to use the properties to create the **ADOR.Recordset** object.

You can also request the type of **ADOR.Recordset** to be created by setting the **GridEX** control's **RecordsetType** property. If you don't request a specific type, a Keyset-type cursor is created. Using the **RecordsetType** property, you can request to create either a Static-, or Keyset-type cursor. However, if the control can't create the type requested, an error event occurs.

If you create an **ADOR.Recordset** object using code, you can set the **ADORecordset** property of the control to this new **ADOR.Recordset**. Any existing **ADOR.Recordset** in the control are released when a new **ADOR.Recordset** is assigned to the **ADORecordset** property.

If you set this property at run time the **DataMode** property of the control is changed to **jpgexADO** no matter the previous **DataMode** property of the control. However if you try to read the **ADORecordset** property when the **DataMode** property of the control is other than **jpgexADO** a trappable error occurs.

Note You must avoid to use the **Close** method in any variable set to **ADORecordset** object because it will make invalid this object for any further operations until the **Rebind** method is used.

Data Type

Object

AfterColEdit Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after data in the current cell is edited.

Syntax

Private Sub *object*_**AfterColEdit**([*index* **As Integer**,] **ByVal** *colindex* **As Integer**)

The **AfterColEdit** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>colindex</i>	An integer that identifies the index of the column that was edited.

Remarks

When the user completes editing within a cell, the **BeforeColUpdate** and **AfterColUpdate** events are executed, and data from the cell is moved to the control's copy buffer. The **AfterColEdit** event immediately follows the **AfterColUpdate** event.

When editing is completed in a cell, this event is only triggered when changes were made to the cell and the **BeforeColUpdate** event was not canceled.

The **AfterColEdit** event will not be fired if the **BeforeColEdit** event is canceled.

AfterColMove Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after the user has dropped a column into a new position.

Syntax

Private Sub *object*_**AfterColMove** ([*index* **As Integer**])

The **AfterColMove** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>Index</i>	An integer that identifies a control if it is in a control array.

Remarks

The event is triggered after the position of a column has changed in response to a user action. The column dragged either was originally in the column headers row or was in the group by box the event will be the same.

The event is preceded by the **BeforeColMove** event and will be fired only if this event is not canceled.

The **AfterColMove** event is fired only if the **View** property setting is **lgexTable**.

AfterColUpdate Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after data is moved from a cell in the **GridEX** control to the control's copy buffer.

Syntax

Private Sub *object*_**AfterColUpdate**([*index* **As Integer**,] **ByVal** *colindex* **As Integer**)

The **AfterColUpdate** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>colindex</i>	An integer that identifies the index of the column updated.

Remarks

When a user completes editing in a cell of the **GridEX** control, the **BeforeColUpdate** event is executed, and unless canceled, data from the cell is moved to the control's copy buffer. Once moved, the **AfterColUpdate** event is executed.

The **AfterColUpdate** event occurs after the **BeforeColUpdate** event only if the *cancel* argument in the **BeforeColUpdate** event is not set to **True**.

Once the **AfterColUpdate** event procedure begins, the cell data has already been moved to the control's copy buffer and can't be canceled, but other updates can occur before the data is committed to the **Recordset**.

AfterDelete Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after the user deletes the selected record in the **GridEX** control.

Syntax

Private Sub *object*_**AfterDelete**([*index* **As Integer**])

The **AfterDelete** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.

Remarks

When the user selects a record selector in the **GridEX** control and presses DEL, the selected row is deleted. Before the record is deleted, the **BeforeDelete** event is triggered and, unless canceled, the **GridEX** will try to delete the row in the **Recordset**. Once the row is deleted, the **AfterDelete** event is triggered.

AfterGroupChange Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after the user has changed, added or removed a group in the **GridEX** Control.

Syntax

Private Sub *object***_AfterGroupChange([** *index* **As Integer])**

The **AfterGroupChange** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.

Remarks

Actions made by the user, like dragging a column header into the group by box, changing the position of a group, or dragging a group outside the group by box, changes the group settings. The **BeforeGroupChange** event is fired when group settings are about to change in response to a user action and, unless canceled, the control will group the rows with the new settings. Done this, the **AfterGroupChange** event will be fired.

Note: The **AfterGroupChange** event will not be fired if the group settings are changed in code.

AfterUpdate Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after changes made by the user have been written to the database from a **GridEX** control.

Syntax

Private Sub *object*_**AfterUpdate**([*index* **As Integer**])

The **AfterUpdate** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.

Remarks

When the user moves to another row, or the Group or Sort is refreshed, data from the **GridEX** control's copy buffer is written to the database. Once the write is complete, the **AfterUpdate** event is triggered.

The **AfterUpdate** event occurs after the **BeforeUpdate** event and only if the **BeforeUpdate** event is not cancelled.

AllowAddNew Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the user can add new records to the **Recordset** object underlying a **GridEX** control.

Syntax

object.**AllowAddNew** [= *value*]

The **AllowAddNew** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether a user can add new records, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Users can add records to the Recordset object underlying the GridEX control.
False	Users can't add records to the Recordset object underlying the GridEX control.

Remarks

If the **AllowAddNew** property is **True**, the **GridEX** control allows the users to enter new records, either in top new row or in last row according to the **NewRowPos** property. If the **AllowAddNew** property is **False**, neither blank line nor top new row is displayed.

The underlying **Recordset** may not enable insertions even if the **AllowAddNew** property is **True**. In this case, the control will act as if the **AllowAddNew** property was set to **False**.

Data Type

Boolean

AllowCardSizing Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the user can resize cards in a **GridEX** control.

Syntax

object.**AllowCardSizing** [= *value*]

The **AllowCardSizing** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether a user can size cards, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Users can resize cards interactively in a GridEX control while in card view.
False	Users can't resize cards in a GridEX control while in card view.

Remarks

If the **AllowCardSizing** property is **True**, the **GridEX** control allows the users to change the width of the cards dragging a bar between cards while the control is in card view.

Data Type

Boolean

AllowDelete Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the user can delete records from the **Recordset** object underlying a **GridEX** control.

Syntax

object.**AllowDelete** [= *value*]

The **AllowDelete** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether a user can delete records, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Users can delete records from the Recordset object underlying the GridEX control.
False	Users can't delete records from the Recordset object underlying the GridEX control.

Remarks

Use the **AllowDelete** property to prevent the user from deleting records from the **Recordset** through interaction with the **GridEX** control.

The underlying **Recordset** may not enable deletions even if the **AllowDelete** property is **True** for the **GridEX** control. In this case, the control will act as if the **AllowDelete** property was set to **False**. If an error occurs when the user tries to delete a record, the **Error** event is triggered.

Data Type

Boolean

AllowEdit Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether a user can modify any data in the **GridEX** control.

Syntax

object.AllowEdit [= *value*]

The **AllowEdit** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether the user can change data, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The user can modify data in the GridEX control
False	The user can't modify data in the GridEX control

Remarks

When the **AllowEdit** property is **False**, the user can still scroll through the **GridEX** control and select data, but can't change any of the values; any attempt to change the data in the control is ignored.

You can also use the **Column** object properties to make individual columns of the **GridEX** control read-only, but the **AllowEdit** property setting takes precedence over the **Column** settings (without changing the **Column** settings).

Note: The **Recordset** object may not enable edits even if **AllowEdit** is **True** for the **GridEX** control; in this case, the control acts as if the **AllowEdit** property was set to **False**. If the **Recordset** is updatable and for some reason the record edited could not be updated an **Error** event is fired when the user tries to change the record.

Data Type

Boolean

AllowColumnDrag Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the user can drag columns or groups in a **GridEX** control.

Syntax

object.**AllowColumnDrag** [= *value*]

The **AllowColumnDrag** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether a user can drag columns or groups while a GridEX control is in table view, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Users can drag columns while a GridEX control is in table view.
False	Users can't drag columns in the GridEX control.

Remarks

If the **AllowColumnDrag** property is **True**, the **GridEX** control allows the users to drag either a column or a group, to change the column position and/or the group settings.

The **AllowColumnDrag** property has no effect if the **View** property setting is set to **jpgexCard**.

Data Type

Boolean

AutomaticArrange Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the **GridEX** control will reposition sorted or grouped records after the user has updated the data in a record.

Syntax

object.**AutomaticArrange** [= *value*]

The **AutomaticArrange** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether the updated record will be repositioned, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The GridEX control will change the position and group of the updated record.
False	The GridEX control will not change the position and group of the updated record.

Remarks

If the **AutomaticArrange** property is **True**, the **GridEX** control will change the position and/or group of a record if necessary. If the **AutomaticArrange** property setting is **False** the edited record will remain in its position until the request of a new sort or group operation.

Even if **AutomaticArrange** property setting is **True**, the control will not reposition records when changes were made anywhere except through user interaction with the **GridEX** control. If you change a record in the **Recordset** clone, and you want to update the position and group for the record, you must set the record updated as the current row and then calls the **GridEX** control's **Update** method.

Data Type

Boolean

BackColor, BackColorBkg, BackColorHeader, BackColorGBBox, BackColorInfoText, BackColorRowGroup Properties

[See Also](#)

[Example](#)

[Applies To](#)

Return or set the background color for elements in a **GridEX** control.

Syntax

object.**BackColor** [=*color*]

object.**BackColorBkg** [=*color*]

object.**BackColorHeader** [=*color*]

object.**BackColorGBBox** [=*color*]

object.**BackColorInfoText** [=*color*]

object.**BackColorRowGroup** [=*color*]

The **BackColor**, **BackColorBkg**, **BackColorHeader**, **BackColorGBBox**, **BackColorInfoText** and **BackColorRowGroup** property syntax have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A numeric expression that specifies the color.

Remarks

The **BackColor** property, in table view, refers to the background color of all the cells. In card view, refers to the background color of the card body.

The **BackColorBkg** property, in table view, refers to the background color of the space without cells. In card view refers to the background color of the space around the cards.

The **BackColorHeader** property in table view refers to the background color of the column and row headers while in card view refers to the background color of the card title bar.

The **BackColorGBBox** property refers to the background color of the group by box. The **BackColorGBBox** is not used in card view.

The **BackColorInfoText** property in table view refers to the background color of the rectangle surrounding the information text displayed in the group by box when the **GridEX** control has no groups. If the **GroupByBoxInfoText** property is a zero length string, the **GridEX** control will not display any rectangle with this color as its background color.

The **BackColorRowGroup** property refers to the background color of the group rows in table view. The **BackColorRowGroup** is not used in card view.

Data Type

OLE_COLOR

BeforeColEdit Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs just before the user enters edit mode by typing a character or clicking in a cell.

Syntax

Private Sub *object*_**BeforeColEdit**([*index* **As Integer**,] **ByVal** *colindex* **As Integer**, *cancel* **As Boolean**)

The **BeforeColEdit** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>Index</i>	An integer that identifies a control if it is in a control array.
<i>colindex</i>	An integer that identifies the index of the column to be edited.
<i>cancel</i>	An integer that may be set to True to prevent the user from editing the cell, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	The cell will not enter edit mode
False	(Default) The cell will enter edit mode and the edit window will appear

Remarks

Use this event to control which cell can be edited, on a per-cell basis. If you want to prevent edition on a per-column basis set the **EditType** property of the column to **kgexEditNone**.

BeforeColMove Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs before repainting the **GridEX** control after the user has dropped a column into a new position.

Syntax

```
Private Sub object_BeforeColMove( [ index As Integer, ] column As JSGridEX.Column, ByVal newposition As Integer, cancel As Boolean)
```

The **BeforeColMove** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>column</i>	A JSGridEX.Column object that identifies the column that is going to be moved.
<i>newposition</i>	An integer that indicates the new position of the column.
<i>cancel</i>	A Boolean that maybe set to True , to prevent the user to move the column as described in settings.

Settings

The settings for *cancel* are:

Setting	Description
True	The column will not be moved to the new position.
False	(Default) The column will be moved to the new position.

Remarks

The event is triggered after the user has dropped a column into a new position but before, the **GridEX** control has been repainted. The column dragged either was originally in the column headers row or was in the group by box, the event will be the same.

When this event is fired, the **ColPosition** property of the **Column** object hasn't been changed, so you can see the original position of the column that is being moved using the **ColPosition** property in the column parameter.

If the *cancel* parameter is set to **True**, the **AfterColMove** event will not be fired

The **BeforeColMove** event is only fired if the control is in table view.

BeforeColumnDrag Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs before the user begins a drag operation with a column header in the column headers row.

Syntax

Private Sub *object***_BeforeColumnDrag ([** *index* **As Integer,]** *column* **As JSGridEX.Column,** *cancel* **As Boolean)**

The **BeforeColumnDrag** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>column</i>	A JSGridEX.Column object that identifies the column that is going to be dragged.
<i>cancel</i>	A Boolean that maybe set to True , to prevent the user to begins the dragging of a column as described in settings.

Settings

The settings for *cancel* are:

Setting	Description
True	The column can't be dragged by the user.
False	(Default) The user can drag the column.

Remarks

The event is triggered before the dragging of a column in the column headers row.

Use this event in order to prevent the interactive moving by user in a per-column basis. If you want to prevent dragging in all the columns, use the **AllowColumnDrag** property instead.

If the column that is about of being dragged is in the group by box this event is not fired, instead the **BeforeGroupDrag** event is fired.

BeforeColUpdate Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after editing is completed in a cell, but before data is moved from the cell to the **GridEX** control's copy buffer.

Syntax

Private Sub *object*_**BeforeColUpdate**([*index* **As Integer**,] **ByVal** *row* **As Long**, **ByVal** *colindex* **As Integer**, *oldvalue* **As String**, *cancel* **As Boolean**)

The **BeforeColUpdate** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>row</i>	A Long that identifies the row position.
<i>colindex</i>	An integer that represents the index of the column.
<i>oldvalue</i>	A String that contains the value contained in the cell prior to the change.
<i>cancel</i>	A Boolean expression that specifies whether the change occurs, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Cancels the change, restores cell to oldvalue.
False	(Default) Continues with the update operation.

Remarks

The data moves from the cell to the control's copy buffer when a user completes editing within a cell, as when tabs to another column in the same row. Before the data has been moved from the cell into the control's copy buffer, the **BeforeColUpdate** event is triggered. This event gives your application an opportunity to check the individual control cells before they are committed to the control's copy buffer.

If your event procedures set the *cancel* argument to **True**, the previous value is restored in the cell and the **AfterColUpdate** event is not triggered.

The **AfterColUpdate** event occurs after the **BeforeColUpdate** event.

BeforeDelete Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs before a selected record is deleted in a **GridEX** control.

Syntax

Private Sub *object*_**BeforeDelete** ([*index* **As Integer**,] *cancel* **As Boolean**)

The **BeforeDelete** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>cancel</i>	A Boolean expression that determines whether a record is deleted, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	The GridEX control doesn't delete the record.
False	(Default) Continues with the delete operation.

Remarks

When the user selects a record selector in the control and presses DEL, the **BeforeDelete** event is triggered before the selected row is deleted.

Once the row is deleted, the **AfterDelete** event is fired.

If your event procedure sets the *cancel* argument to **True**, the row isn't deleted and the **AfterDelete** event isn't fired.

BeforeGroupChange Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs after the user has changed, added or removed a group in the **GridEX** control but before the change is committed.

Syntax

```
Private Sub object_BeforeGroupChange( [ index As Integer,] group As JSGridEX.Group, ByVal changeoperation As jsgeXGroupChange, ByVal groupposition As Integer, cancel As Boolean)
```

The **BeforeGroupChange** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>group</i>	A JSGridEX.Group object that identifies the group that is about to change.
<i>changeoperation</i>	A value or constant that identifies the type of change in the group as described in settings.
<i>groupposition</i>	An integer that indicates the position of the changing group.
<i>cancel</i>	A Boolean expression that determines whether the changing operation will be committed as described in Settings.

Settings

The settings for *changeoperation* are:

Constant	Setting	Description
jgexGroupInsert	0	A column dragged from the column header row was dropped in the group by box and is about of being inserted in the groups collection.
jgexGroupDelete	1	A column dragged from the group by box was dropped outside the control or in the column header row and is about of being deleted from the groups collection.
jgexGroupMove	2	A column dragged from the group by box was dropped in the group by box again but in a new position for the group.

The settings for *cancel* are:

Setting	Description
True	The group change operation is stopped and the group settings remain the same.
False	(Default) The group change operation will be committed.

Remarks

Whenever the changes the group settings, as when the user drags a column into the group by box, change the position of a group, or drags a group outside the group by box, the **BeforeGroupChange** event is fired. Unless canceled the control will group the rows with the new settings, done this, the **AfterGroupChange** event is fired.

The **BeforeGroupChange** will not be fired if the group settings are changed in code.

BeforeGroupDrag Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs before the user begins a drag operation with a column header in the group by box.

Syntax

Private Sub *object***_BeforeGroupDrag([** *index* **As Integer,]** *group* **As JSGridEX.Group,** *cancel* **As Boolean)**

The **BeforeGroupDrag** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>group</i>	A JSGridEX.Group object that identifies the group that is going to be dragged.
<i>cancel</i>	A Boolean that maybe set to True , to prevent the user to begins the dragging of a group as described in settings.

Settings

The settings for *cancel* are:

Setting	Description
True	The group header can't be dragged by the user.
False	(Default) The user can drag the group header.

Remarks

The event is triggered before the dragging of a group begins.

Use this event in order to prevent the interactive moving by the user. If you want to prevent dragging in all the columns and all groups use the **AllowColumnDrag** property instead.

If the column header that is about of being dragged is in the column header row this event is not fired, the **BeforeColumnDrag** event is fired instead.

BeforeUpdate Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs before data is committed from a **GridEX** control to the database.

Syntax

Private Sub *object***_BeforeUpdate([** *index* **As Integer,]** *cancel* **As Boolean)**

The **BeforeUpdate** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>cancel</i>	A Boolean expression that determines if data is written to the database, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	The GridEX doesn't write changes to the database.
False	(Default) Continues with the update operation.

Remarks

When the user moves to another row or the control is about to change sort or group settings, the **Update** method is executed and data from the **GridEX** control's copy buffer is written to the database.

Just before the changes are committed to the database, the **BeforeUpdate** event is triggered. Unless the update operation is canceled, the **AfterUpdate** event is fired after the data has been written to the database.

If you set the **BeforeUpdate** event *cancel* argument to **True**, the **AfterUpdate** event will not be fired, and the record isn't saved to the database.

The **BeforeUpdate** event occurs before the **AfterUpdate** for this control.

You can use this event to validate data in the current record before permitting the user to commit the change to the database. Setting the *cancel* argument to **True**, the user will not be able to move to another record in the control. If you want to restore the record values and cancel the update operation, you can set the **GridEX** control's **DataChanged** property to **False**.

BorderStyle Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the border style for an object. Read-only at run time.

Syntax

object.**BorderStyle** [= *value*]

The **BorderStyle** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the border style, as described in Settings.

Settings

The **BorderStyle** property settings are:

Constant	Setting	Description
jgexNone	0	None (no border).
jgexFixed	1	Fixed. The control appears with a 3D border.

Remarks

If the **BorderStyle** property is set to **jgexFixed**, the control appears with a 3D sunken border.

Data Type

Integer

CardBorders Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets whether the **GridEX** will show borders in cards while in card view.

Syntax

object.**CardBorders** [= *value*]

The **CardBorders** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines the whether the GridEX control will show borders in cards, as described in Settings.

Settings

The **CardBorders** property settings are:

Setting	Description
True	The cards in GridEX control will appear with borders.
False	The card in GridEX control will appear with no borders.

Data Type

Boolean

CardSpacing Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the horizontal and vertical space between cards.

Syntax

object.**CardSpacing** [=*value*]

The **CardSpacing** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Long expression that specifies the space, in pixels, between cards while the GridEX is in card view.

Remarks:

The **CardSpacing** property is given in pixels.

Data Type

Long

CardCaptionPrefix Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the text that will appear before the caption in every card.

Syntax

object.**CardCaptionPrefix** [= *value*]

The **CardCaptionPrefix** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that specifies the text that will appear before the caption in every card.

Data Type

String

CardWidth Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the width of a card while the **GridEX** is in card view.

Syntax

object.**CardWidth** [= *value*]

The **CardWidth** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Long expression that specifies the width, in twips, of every card while the GridEX control is in card view.

Remarks:

The **CardWidth** property is given in twips.

Data Type

Long

ClearFields Method

[See Also](#)

[Example](#)

[Applies To](#)

Restores the default **GridEX** control's column layout.

Syntax

object.**ClearFields**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **ClearFields** method restores the default column layout (two blank columns) so that, subsequent **ReBind** operations will automatically derive new column bindings from the (possibly changed) data source. You can cancel the **GridEX** control's automatic layout behavior by invoking the **HoldFields** method.

Note If you want to clear all the columns in a **GridEX** control, use the **Clear** method in the **Columns** collection.

Click Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when the user presses and then releases a mouse button over an object

Syntax

Private Sub *object_Click*([*index* **As Integer**])

The **Click** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that uniquely identifies a control if it's in a control array.

Remarks

Clicking the **GridEX** control, generates **MouseDown** and **MouseUp** events in addition to the **Click** event. The events occur in this order: **MouseDown**, **MouseUp**, and **Click**. When you're attaching event procedures for these related events. Be sure that their actions don't conflict.

Note To distinguish between the left, right, and middle mouse buttons, use the **MouseDown** and **MouseUp** events.

Col Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the active column in a **GridEX** control. Not available at design time.

Syntax

object.**Col** [= *value*]

The **Col** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	An integer that represents the position of the column containing the active cell.

Remarks

Use this property to specify a cell in a **GridEX** control or to find out which column contains the active cell. In the **GridEX** control, columns and rows are numbered from 1, beginning at the left for columns.

This property represents the column position and not its index. Because the columns' position could be changed, the index and the position of a column are not always the same. If you want to find out what index has the current column you may do something like this:

```
Dim Col as JSGridEX.Column
Dim ColPosition as Integer
Dim ColIndex as Integer
    ColPosition = GridEX1.Col
    Set Col = GridEX1.Columns.ItemByPosition(ColPosition)
    ColIndex = Col.Index
    Debug.Print ColIndex
```

A value of 0 in the **Col** property means that the entire row is selected.

If you set the **Col** property with a value greater than the visible columns count, the **GridEX** control will select the last column visible.

This property could be used in both, table and card, views. The only difference between views is that, in table view the **Col** property represents a column in the current row while in card view represents a row field in the current card.

ColumnHeaderClick Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when the user clicks on the header for a particular **Column** of a **GridEX** control while it's in table view.

Syntax

Private Sub *object*_**ColumnHeaderClick**([*index* **As Integer**,] *column* **As JSGridEX.Column**)

The **ColumnHeaderClick** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>column</i>	A JSGridEX.Column object that identifies the column.

Remarks

One possible use for this event, is to sort the records in the **GridEX** control based on the selected column. While the control groups automatically, based on user interaction, the sort of records has to be done by the programmer. A standard procedure that does this could be like this:

```
Private Sub GridEX1_ColumnHeaderClick(Column As JSGridEX.Column)
Dim SortOrder as integer
    SortOrder = Column.SortOrder
    GridEX1.SortKeys.Clear
    If SortOrder = jgexSortAscending Then
        GridEX1.SortKeys.Add Column.Index, jgexSortDescending
    Else 'if SortOrder is none or is descending
        GridEX1.SortKeys.Add Column.Index, jgexSortAscending
    End If
End Sub
```

ColResize Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when a user resizes a column of a **GridEX** control.

Syntax

Private Sub *object*_**ColResize**([*index* **As Integer**,] **ByVal** *colindex* **As Integer**, *cancel* **As Boolean**)

The **ColResize** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>colindex</i>	An integer that represents the index of the column that is about to be resized.
<i>cancel</i>	A Boolean expression that determines whether a column is resized, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Cancels the change, restores column to its original width.
False	(Default) Continues with width change.

Remarks

When the user resizes a column, the **ColResize** event is triggered. Your event procedure can accept the change, alter the degree of change, or cancel the change completely.

If you set the *cancel* argument to **True**, the column width is restored. To alter the degree of change, set the **Width** property of the **Column** object to the desired value.

ColumnHeaderFont Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating the font used in column headers in a **GridEX** control.

Syntax

object.**ColumnHeaderFont** [= *font*]

The **ColumnHeaderFont** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>font</i>	An object expression that evaluates to a Font object.

Remarks

The **ColumnHeaderFont** is used in column headers as well as in the group by box information text, if the control is in table view. In card view, the **ColumnHeaderFont** property represents the font used in the caption bar of a card.

Changing the **ColumnHeaderFont** property may resize the headers to accommodate the new font.

Data Type

Font

ForeColor, ForeColorHeader, ForeColorInfoText, ForeColorRowGroup Properties

[See Also](#)

[Example](#)

[Applies To](#)

Return or set the foreground color, for elements of the **GridEX**, used to display text.

Syntax

object.**ForeColor** [=*color*]

object.**ForeColorHeader** [=*color*]

object.**ForeColorInfoText** [=*color*]

object.**ForeColorRowGroup** [=*color*]

The **ForeColor**, **ForeColorHeader**, **ForeColorInfoText**, **ForeColorRowGroup** property syntax have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A numeric expression that specifies the color.

Remarks

The **ForeColor** property refers to the text color of all the cells in table and card view.

The **ForeColorHeader** property in table view refers to the text color of the column and row headers while in card view refers to the text color of the card caption.

The **ForeColorInfoText** property refers to the text color of the information text displayed in the group by box when there is no groups, in table view. The **ForeColorInfoText** is not used in card view.

The **ForeColorRowGroup** property refers to the background color of the group rows in table view. The **ForeColorRowGroup** is not used in card view.

Data Type

OLE_COLOR

ColumnHeaderHeight Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the height of the column header row in a **GridEX** control.

Syntax

object.**ColumnHeaderHeight** [= *value*]

The **ColumnHeaderHeight** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Long expression that specifies the height of the column header row in a GridEX control.

Remarks

The **ColumnHeaderHeight** property is calculated every time the **ColumnHeaderFont** changes to accommodate the new **Font**. However, if you want another height than the default height you can set this property to any valid dimension.

The **ColumnHeaderHeight** property is given in twips.

Data Type

Long

ColumnHeaders Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the column headers are displayed in a **GridEX** control.

Syntax

object.**ColumnHeaders** [= *value*]

The **ColumnHeaders** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether column headers are displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	GridEX control's column headers are displayed
False	GridEX control's column headers aren't displayed

Remarks:

When the **GridEX** control is in card view this property has no effect.

Columns Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **Columns** collection in a **GridEX** control.

Syntax

object.**Columns**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **Columns** property returns a **Columns** object. The **Columns** object is a collection of **Column** objects in a **GridEX** control. You can get a specific column through the **Item** property or through the **ItemByPosition** method.

You can manipulate most of a **GridEX** control's attributes by changing the properties of **Column** objects.

Data Type

Columns

Connect Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that indicates to a **GridEX** control the *connect* parameter used to open the database when **DataMode** Property is set to **kgexDAO**.

Syntax

object.**Connect** [= *value*]

The **Connect** property syntax has these parts.

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A String that specifies the Connect property of the database for use when its DataMode property is set to kgexDAO .

Remarks

This property has only effect when the **DataMode** property setting is **kgexDAO**.

Data Type

String

ContinuousScroll Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that determines whether **GridEX** control should scroll its contents while the user moves the scroll box along the vertical scroll bar.

Syntax

object.**ContinuousScroll** [= *boolean*]

The **ContinuousScroll** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>boolean</i>	A Boolean expression that determines if scrolling is made when the user drags the scroll box. This property is False by default.

Remarks

This property should normally be set to **False** to avoid excessive scrolling and flickering. Set it to **True** only if you want to emulate other controls that have this behavior, or when you want to view rows as they are being scrolled.

Note This property has effect, only in table view.

Data Type

Boolean

Change Event

[See Also](#)

[Example](#)

[Applies To](#)

Indicates the contents of the current cell have changed.

Syntax

Private Sub *object_Change*([*index* **As Integer**])

The **Change** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that uniquely identifies a control if it's in a control array

DatabaseName Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the name and location of the source of data for a **GridEX** control.

Syntax

object.**DatabaseName** [= *name*]

The **DatabaseName** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>name</i>	A string expression that indicates the location of the database file(s) or the Data Source Name when the control's DataMode is ADO.

Remarks

If you change the **DatabaseName** property after the control's **Database** object is open, you must use the **Rebind** method to open the new database.

When a **GridEX** Control **DataMode** property setting is **jpgxDAO**, the **DatabaseName** represents the *name* parameter of the **OpenDatabase** method. In ADO mode, it represents the **ActiveConnection** string of an **ADOR.Recordset**

Data Type

String

DataChanged Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating that the data in a **GridEX** control has been changed by some process other than that of retrieving data from the current record. Not available at design time.

Syntax

object.DataChanged [= *value*]

The **DataChanged** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that indicates whether data has changed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The data currently in the control isn't the same as in the current record.
False	(Default) The data currently in the control, if any, is the same as the data in the current record.

Remarks

When the user has made changes to current record in a **GridEX** control, the **DataChanged** property is set to **True** and before the user moves to another record the control will try write those changes in the database.

If you don't wish to save changes to the database, you can set the **DataChanged** property to **False**.

Data Type

Boolean

DataMode Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value representing the **GridEX** data retrieval mode. Read-only at run time.

Syntax

object.**DataMode** [= *value*]

The **DataMode** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the data retrieval mode in a GridEX control, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
jgexDAO	0	The GridEX uses DAO to create the Recordset .
jgexADO	1	The GridEX uses ADO to create the Recordset .
jgexUnbound	99	The GridEX uses the unbound events to retrieve and update displayed data.

Remarks:

The **DataMode** property controls how the data is handled for the **GridEX** control. In unbound mode, you are responsible for maintaining data and supplying the **GridEX** control with the appropriate data when requested through the unbound events. In DAO and ADO mode, the data is retrieved and updated automatically using the **Recordset** created by the control.

Normally, the unbound mode of the **GridEX** control is used when displaying data that is not stored in a database accessible by DAO or ADO. You can use the unbound mode for whatever type of data you have available. For example, you can use the unbound mode of the **GridEX** control to display data from a proprietary database format or use it to manage data that you keep track of in a text file.

Note In unbound mode there is a performance penalty in sorting and grouping compared with the others modes.

Data Type

Integer

DbClick Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when the user presses and releases a mouse button two times over a **GridEX** control.

Syntax

Private Sub *object*_**DbClick**([*index* **As Integer**])

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	Identifies the control if it's in a control array.

Remarks

The mouse events in a **GridEX** control occur in this order: **MouseDown**, **MouseUp**, **Click**, **DbClick**, and **MouseUp**.

If **DbClick** doesn't occur within the system's double-click time limit, the object recognizes another **Click** event.

Note: To distinguish between the left, right, and middle mouse buttons, use the **MouseDown** and **MouseUp** events.

DefaultColumnWidth Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating the default column width used for new columns, in a **GridEX** control.

Syntax

object.**DefaultColumnWidth** [= *value*]

The **DefaultColumnWidth** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Long expression that identifies the default width, in twips, for columns.

Remarks

The **DefaultColumnWidth** property is given in twips.

Data Type

Long

EnsureVisible Method

[See Also](#)

[Example](#)

[Applies To](#)

Ensures visibility of a particular cell. If necessary, this method scrolls **GridEX** control.

Syntax

object.**EnsureVisible**(*row*, *col*)

The **EnsureVisible** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>row</i>	(Optional) A long expression that identifies the row, if omitted; the current row is used.
<i>col</i>	(Optional) An integer expression that identifies the column, if omitted; the current column is used.

Remarks

Use the **EnsureVisible** method when you want a particular cell, which might be hidden, to be visible.

Sometimes the current cell could be hidden because the user has scrolled the control through the scrollbar. If you want to ensure the visibility of the current cell, use this method without parameters.

Error Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs as the result of a data access error.

Syntax

Private Sub *object_Error*([*index As Integer*,] **ByVal** *errnumber As Long*, *displaymessage As Boolean*)

The **Error** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>Index</i>	An integer that identifies a control if it is in a control array.
<i>errnumber</i>	A Long that identifies the error that occurred.
<i>displaymessage</i>	A Boolean that may be set to False to suppress error message display, as described in Settings.

Settings

The settings for *displaymessage* are:

Setting	Description
False	No error message will be displayed.
True	(Default) The message associated with the error will be displayed.

Remarks

Even if your application handles run time errors in code, errors can still occur when none of your code is executing. For example is when the **GridEX** control tries to create a **Recordset** and one or more properties had been set incorrectly. Another example is when the **GridEX** control tries to write changes made by the user into a record that is locked by another user. If a data access error results from such an action, the **Error** event is fired.

You may use the **Error** event in order to reset changes made to the current record that had caused that error.

Note Use the **ErrorText** property to retrieve or modify the error string that will be displayed.

ErrorText Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the error message String from the underlying data source. Not available at design time.

Syntax

object.**ErrorText** [= *value*]

The **ErrorText** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string that represents the text that will be displayed if an error occurs.

Remarks

When a database error occurs as a result of user interaction with the control, such as when the user enters text into a numeric field and then attempts to update the current, the control's **Error** event is fired. However, the error code passed to the event handler in the *errnumber* parameter may not identify the specific error that occurred, or may even differ across operating environments. For these reasons, the **ErrorText** property is provided so that your application can parse the actual error message to determine the nature of the error.

If you want to customize the error description displayed by the control you may set the **ErrorText** property to any string or cancel the default response in the error event and display your custom message.

Note The **ErrorText** property is only valid within a **GridEX** control's **Error** event handler.

Exclusive Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that indicates whether the underlying database for a **GridEX** control is opened for single- or multi-user access when DAO mode is specified.

Syntax

object.**Exclusive** [= *value*]

The **Exclusive** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines user access, as described in Settings.

Settings

The settings for *value* are:

Part	Description
True	The database is open for single-user access. No one else can open the database until it's closed.
False	(Default) The database is open for multi-user access. Other users can open the database and have access to the data while it's open.

Remarks

This property only affects the way that **GridEX** open a database in DAO mode.

The value of this property, along with the **DatabaseName**, **ReadOnly**, and **Connect** properties, is used to open a database. This property corresponds to the exclusive argument in the **OpenDatabase** method.

The **Exclusive** property is used only when opening the Database. If you change the value of this property at run time, you must use the **Rebind** method for the change to take effect. If someone else already has the database open, you can't open it for exclusive use and an **Error** event is fired.

FetchData Event

[See Also](#)

[Example](#)

[Applies To](#)

Fetches unbound column data when the **GridEX** is in DAO or ADO mode.

Syntax

Private Sub *object***_FetchData([** *index* **as Integer,]** **ByVal** *rowindex* **As Long,** **ByVal** *colindex* **As Integer,** **ByVal** *rowbookmark* **as Variant,** *value* **as Variant)**

The **FetchData** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>Index</i>	An integer that identifies a control if it is in a control array.
<i>rowindex</i>	A long that identifies the index of the record being fetched.
<i>colindex</i>	An integer that identifies the index of the column being fetched.
<i>rowbookmark</i>	A variant that identifies the bookmark of the record being fetched.
<i>value</i>	A variant expression that represents the value of the cell being fetched.

Remarks:

This event is only fired for the columns whose property **FetchData** is set to **True**.

The *rowindex* parameter represents the index of the record as it was originally in the **Recordset**. Because the position of a record could be changed if the control is sorted or grouped, the row index will not always match with its position. The *colindex* parameter also represents the index of the column and not its position in that way you can write the event procedure without worrying if the user has changed the column position.

The *rowbookmark* parameter could be used if you want to access the record values via the **Recordset** or the **ADORecordset** property. A **FetchData** event procedure could be like this:

```
Private Sub GridEX1_FetchData(ByVal RowIndex As Long, _
                               ByVal ColIndex As Integer, _
                               ByVal RowBookmark as Variant, _
                               Value as Variant)

    dim rsTemp as Recordset
    Set rsTemp=GridEX1.Recordset
    rsTemp.Bookmark = RowBookmark

    'The value is given adding the value of field 1
    'to the value of field 3
    Value = rsTemp.Fields(1).Value + rsTemp.Fields(3).Value
End Sub
```

FetchIcon Event

[See Also](#)

[Example](#)

[Applies To](#)

Fetches icons for any column whose **FetchIcon** property is set to **True**.

Syntax

Private Sub *object***_FetchIcon([** *index* **as Integer,] ByVal** *rowindex* **As Long, ByVal** *colindex* **As Integer, ByVal** *rowbookmark* **As Variant,** *iconindex* **as Integer)**

The **FetchIcon** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>rowindex</i>	A long that identifies the Index of the record being fetched.
<i>colindex</i>	An integer that identifies the index of the column being fetched.
<i>rowbookmark</i>	A variant that identifies the bookmark of the record being fetched.
<i>iconindex</i>	An integer expression that specifies the icon index of the cell being fetched.

Remarks:

This event is only fired for the columns whose property **FetchIcon** is set to **True**.

The *rowindex* parameter represents the index of the record as it was originally in the **Recordset**. Because the position of a record could be changed if the control is sorted or grouped, the row index will not always match with its position. The *colindex* parameter also represents the index of the column and not its position in that way you can write the event procedure without worrying if the user has changed the column position.

The *rowbookmark* parameters could be used if you want to access the record values via the **Recordset** or the **ADORRecordset** property. A **FetchData** event procedure could be like this:

```
Private Sub GridEX1_FetchIcon (ByValRowIndex As Long, _  
                               ByVal ColIndex As Integer, _  
                               ByVal RowBookmark As Variant, _  
                               IconIndex as Integer)  
  
    Dim rsTemp as Recordset  
    const IconIndexPaid = 1  
    const IconIndexUnPaid = 2  
    set rsTemp = GridEX1.Recordset  
    rsTemp.Bookmark = RowBookmark  
    'The Icon depends of the value of "Paid" Field  
    if rsTemp![Paid] = True then  
        IconIndex = IconIndexPaid
```

```
Else
    IconIndex = IconIndexUnpaid
End if
End Sub
```

FirstItem Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the row position of the first visible row or card, in a **GridEX** control. Not available at design time.

Syntax

object.**FirstItem** [= *value*]

The **FirstItem** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A long expression corresponding to the record position of the first visible row in the GridEX control.

Remarks

For a **GridEX** control, setting the **FirstItem** property causes the control to scroll so that the specified row or card becomes the topmost row or the upper-left card.

When the **GridEX** control is in card view, setting the **FirstItem** property ensures visibility of a card but, depending on its position could be other card the one in upper-left corner.

Data Type

Long

FirstItemChange Event

[See Also](#)

[Example](#)

[Applies To](#)

Fired when the **FirstItem** property changes.

Syntax

Private Sub *object*_**FirstItemChange**([*index* **As Integer**])

The **FirstItemChange** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.

Remarks

Any time the **GridEX** control's first item displayed changes, as when the user scrolls the control, the **FirstItemChange** event is fired.

FmtConditions Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **FmtConditions** object in a **GridEX** control.

Syntax

object.**FmtConditions**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **FmtConditions** property returns a **FmtConditions** collection. You can get a specific **FmtCondition** through the **Item** property in **FmtConditions** object, you can also get a special **FmtCondition** object for expanding it through the group rows.

Font Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a **Font** object used in a **GridEX** control.

Syntax

object.**Font**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Use the **Font** property of an object to identify a specific **Font** object whose properties you want to use. For example, the following code changes the **Bold** property setting of a **Font** object identified by the **Font** property of a **GridEX** control:

```
GridEX1.Font.Bold = True
```

Data Type

Font

GridImages Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **GridImages** collection in a **GridEX** control.

Syntax

object.**GridImages**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **GridImages** property returns a **GridImages** collection. You can get a specific **GridImage** through the **Item** property.

Data Type

GridImages

GridLines Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that determines whether the **GridEX** control will draw lines between cells.

Syntax

object.**GridLines** [= *value*]

The **GridLines** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression specifying whether the GridEX control will draw lines, as described in Settings.

Settings

The settings for *value* are:

Value	Description
False	No Lines. No lines in-between cells.
True	(Default). The GridEX control will draw lines between cells.

Remarks

When the **GridLines** property is set to **True**, the color of the lines is determined by the **GridLinesColor** property. This property has no effect when a **GridEX** control's view is card.

Data Type

Boolean

GridLinesColor Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the color used to draw the lines between the **GridEX** control's cells.

Syntax

object.**GridLinesColor** [= *color*]

The **GridLinesColor** syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A value or constant that determines the color used to paint the grid lines in a GridEX control.

Remarks

The **GridLinesColor** property is used only when the **GridLines** property is set to **True**.

Data Type

OLE_COLOR

GroupByBoxHeaderClick Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when the user clicks on a header group of a **GridEX** control.

Syntax

Private Sub *object_GroupByBoxHeaderClick***([** *index* **As Integer,]** *group* **As** *jSGridEX.Group***)**

The **GroupByBoxHeaderClick** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.
<i>group</i>	A jSGridEX.Group object that identifies the group.

Remarks

One possible use for this event is to change the sort order of the group being clicked in the **GridEX** control. While the control groups automatically based on user interaction, changing the sort of one group has to be done by the programmer. A standard procedure that does this could be like this:

```
Private Sub GridEX1_GroupByBoxHeaderClick(Group As jSGridEX.Group)
    If Group.SortOrder = jgexSortAscending Then
        Group.SortOrder = jgexSortDescending
    Else
        Group.SortOrder = jgexSortAscending
    End If
End Sub
```

GroupByBoxInfoText Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or set the text displayed in the group by box when no groups are applied.

Syntax

object.**GroupByBoxInfoText** [= *value*]

The **GroupByBoxInfoText** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string that represents the text that will be displayed in the group by box when no groups are applied.

Remarks

In order to give some information to the users that they can drag columns into the group by box for grouping records, the **GroupByBoxInfoText** is displayed in the group by box. You can change this property to any string for customization.

This property is used only when the **GridEX** control is in table view.

Data Type

String

GroupByBoxVisible Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the group by box is displayed in a **GridEX** control.

Syntax

object.**GroupByBoxVisible** [= *value*]

The **GroupByBoxVisible** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether group by box is displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	GridEX control's group by box is displayed
False	GridEX control's group by box isn't displayed

Remarks:

When the **GridEX** control is in card view this property has no effect.

Data Type

Boolean

Groups Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **Groups** collection in a **GridEX** control.

Syntax

object.**Groups**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **Groups** property returns a **Groups** collection. You can get a specific group through the **Item** property in the **Groups** collection. You can also add and delete groups.

Data Type

Groups

HoldFields Method

[See Also](#)

[Example](#)

[Applies To](#)

Prevents recalculation of the column layout when the **Recordset** is created.

Syntax

object.**HoldFields**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **HoldFields** method sets the current column/field layout as the customized layout so that subsequent **ReBind** operations will use the current layout for display. You can resume the **GridEX** control's automatic layout behavior by invoking the **ClearFields** method.

hWnd Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the handle of a **GridEX** control.

Syntax

object.**hWnd**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The Microsoft Windows operating environment identifies each control in an application by assigning it a handle, or hWnd. The **hWnd** property is used with Windows API calls. Many Windows operating environment functions require the **hWnd** of the active window as an argument.

Data Type

Long

IsGroupItem Method

[See Also](#)

[Example](#)

[Applies To](#)

Returns **True** if the specified row is a group in a **GridEX** control.

Syntax

object.IsGroupItem(*row*)

The **IsGroupItem** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>row</i>	A long expression that represents the position of the row being tested.

Settings:

The settings for **IsGroupItem** are:

Setting	Description
False	The row specified in the <i>row</i> parameter is not a group row.
True	The row specified in the <i>row</i> parameter is a group row.

Remarks:

When a **GridEX** control is grouped, some rows are group rows and don't represent a record. You may use this method to find out whether the current row is a group or record row.

Note If the control isn't grouped, this method always returns **False**.

Data Type

Boolean

ItemCount Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the number of items (rows) in a **GridEX** control.

For DAO and ADO modes, this property is read only. For unbound mode, this property is read-write.

Syntax:

object.**ItemCount** [= *value*]

The **ItemCount** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A long expression that determines the number of items displayed in a GridEX control.

Remarks:

Because the **GridEX** control loads records until is needed, the **ItemCount** property in DAO and ADO modes could be less than the real count, if the **GridEX** control doesn't have loaded the entire **Recordset**. To see if the entire **Recordset** is loaded you must check the value of **FullyLoaded** property. If you want to force the load of the **Recordset**, use the **LoadEntireRecordset** Method

In unbound mode, the control will only fetch data for the number of rows specified in the **ItemCount** property.

Note Even in unbound mode, you can not set the **ItemCount** property while the control is sorted or grouped.

Data Type

Long

FullyLoaded Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns whether the control has loaded the entire **Recordset**. Read only. Not available at design time.

Syntax

object.**FullyLoaded**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Settings:

The settings for **FullyLoaded** property are:

Setting	Description
False	The GridEX control has not reached the end of the Recordset.
True	The GridEX control has reached the end of the Recordset.

Remarks:

When the **GridEX** control is in unbound mode the **FullyLoaded** property has no meaning and always returns **True**.

The **FullyLoaded** property is read-only. However, if you want to set this property to **True**, you can call the **LoadEntireRecordset** method. An example of this is as follows:

```
If not GridEX1.FullyLoaded then
    GridEX1.LoadEntireRecordset
End if

Debug.Print GridEX1.FullyLoaded 'It will always print true
```

Data Type

Boolean

LoadEntireRecordset Method

[See Also](#)

[Example](#)

[Applies To](#)

Loads the bookmarks of all records in the underlying **Recordset** of a **GridEX** control.

Syntax

object.**LoadEntireRecordset**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **LoadEntireRecordset** method loads all the records in the underlying **Recordset**. This method is only used in DAO and ADO modes. However, invoking this method while the **GridEX** control is in unbound method will not generate any error.

The **GridEX** control loads the **Recordset** only as needed, so this method ensures that all the records has been retrieved.

KeyDown, KeyUp Events

[See Also](#)

[Example](#)

[Applies To](#)

Occur when the user presses (**KeyDown**) or releases (**KeyUp**) a key while an object has the focus.

Syntax

Private Sub *object_KeyDown*([*index As Integer*,] *keycode As Integer*, *shift As Integer*)

Private Sub *object_KeyUp*([*index As Integer*,] *keycode As Integer*, *shift As Integer*)

The **KeyDown** and **KeyUp** event syntax have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that uniquely identifies a control if it's in a control array.
<i>keycode</i>	A integer that represents the key code.
<i>shift</i>	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The <i>shift</i> argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Remarks

Use **KeyDown** and **KeyUp** event procedures if you need to respond to both the pressing and releasing of a key.

You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the **And** operator with the *shift* argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

```
ShiftDown = (Shift And 1) > 0
```

```
CtrlDown = (Shift And 2) > 0
```

```
AltDown = (Shift And 4) > 0
```

In a procedure, you can test for any combination of conditions, as in this example:

```
If AltDown And CtrlDown Then
```

KeyPress Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when the user presses and releases an ANSI key.

Syntax

Private Sub *object*_KeyPress([*index* **As Integer**,] *keyascii* **As Integer**)

The **KeyPress** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that uniquely identifies a control if it's in a control array.
<i>keyascii</i>	An integer that returns a standard numeric ANSI keycode. <i>keyascii</i> is passed by reference; changing it sends a different character to the object. Changing <i>keyascii</i> to 0 cancels the keystroke so the object receives no character.

Remarks

The **KeyPress** event enables you to immediately test keystrokes for validity or formatting of characters as they're typed. Changing the value of the *keyascii* argument changes the character displayed.

Use **KeyDown** and **KeyUp** event procedures to handle any keystroke not recognized by **KeyPress**, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the **KeyDown** and **KeyUp** events, **KeyPress** doesn't indicate the physical state of the keyboard; instead, it passes a character.

KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters.

LeftCol Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the left-most visible column in a **GridEX** control. Not available at design time.

Syntax

object.**LeftCol** [= *value*]

The **LeftCol** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	An Integer expression specifying the left-most column.

Remarks

You can use this property in code to scroll the **GridEX** control programmatically. Use the **FirstItem** property to determine the topmost visible row in the **GridEX** control.

Note If the **GridEX** control is in card view, this property has no use.

LeftColChange Event

[See Also](#)

[Example](#)

[Applies To](#)

Fired when the **LeftCol** property changes.

Syntax

Private Sub *object*_LeftColChange([*index* **As Integer**])

The **LeftColChange** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it is in a control array.

Remarks:

Any time the **GridEX** control's leftmost column displayed changes as when the user scrolls the control, the **LeftColChange** event is fired.

This event is not fired when the **GridEX** control is in card view.

MaskColor Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the color used to create masks for images in a **GridEX** control.

Syntax

object.MaskColor [= *color*]

The **MaskColor** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A value or constant that determines the color used to create masks.

Remarks

Every image in a **GridImages** collection has a corresponding mask associated with it. The mask is a monochrome image derived from the image itself, automatically generated using the **MaskColor** property as the specific color of the mask. This mask is not used directly, but is applied to the original bitmap when the **GridEX** control draws the image. The **MaskColor** property determines which color of an image will be transparent.

It is good programming practice to use non-system colors.

When using icons in **GridImages**, the **MaskColor** property is irrelevant.

Data Type

OLE_COLOR

MouseDown, MouseUp Events

[See Also](#)

[Example](#)

[Applies To](#)

Occur when the user presses (**MouseDown**) or releases (**MouseUp**) a mouse button.

Syntax

Private Sub *object* _**MouseDown**([*index* **As Integer**,] *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single**)

Private Sub *object* _**MouseUp**([*index* **As Integer**,] *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single**)

The **MouseDown** and **MouseUp** event syntax have these parts:

Part	Description
<i>object</i>	Returns an object expression that evaluates to an object in the Applies To list.
<i>index</i>	Returns an integer that uniquely identifies a control if it's in a control array.
<i>button</i>	Returns an integer that identifies the button that was pressed (MouseDown) or released (MouseUp) to cause the event.
<i>shift</i>	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
<i>x, y</i>	Returns a number that specifies the current location of the mouse pointer. The x and y values are always expressed in twips.

Remarks

Use a **MouseDown** or **MouseUp** event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the **Click** and **DbClick** events, **MouseDown** and **MouseUp** events enable you to distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

MouseMove Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when the user moves the mouse.

Syntax

Private Sub *object_MouseMove*([*index As Integer*,] *button As Integer*, *shift As Integer*, *x As Single*, *y As Single*)

The **MouseMove** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that uniquely identifies a control if it's in a control array.
<i>button</i>	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down.
<i>shift</i>	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
<i>x, y</i>	A number that specifies the current location of the mouse pointer. The x and y values are always expressed in twips

Remarks

The **MouseMove** event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a **MouseMove** event whenever the mouse position is within its borders.

NewRowPos Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the position of the row where the user can add new records.

Syntax

object.**NewRowPos** [= *value*]

The **NewRowPos** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the position of the row for new records, as described in Settings.

Settings

The settings for *value* are:

Constant	Setting	Description
jpgexTop	0	The row for adding new records is positioned apart, below the column headers row, in a GridEX control
jpgexBottom	1	The row for adding new records is positioned as last row in the grid.

Remarks:

If the control is in card view, this property has no use because in card view the user can't add records.

If the **AllowAddNew** property is set to **False** or the underlying **Recordset** is not updatable, the **GridEX** control will hide the row for new records no matter the value of this property.

If the **NewRowPos** property is set to **jpgexBottom**, the **GridEX** control will show the row for new records as the last record, except if it's grouped, when no row at all will be displayed.

Data Type

jpgexNewRowPosConstants

Rebind Method

[See Also](#)

[Example](#)

[Applies To](#)

Forces re-creation of the **Recordset** in a **GridEX** control.

Syntax

object.**Rebind**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **Rebind** method causes the **GridEX** control to close the current **Recordset** and create a new one with the parameters specified in the **DatabaseName**, **Exclusive**, **LockType**, **ReadOnly**, **RecordsetType** and **Connect** properties.

If you have not modified the grid columns at design time, then executing the **ReBind** method will reset the columns, headings, and other properties based on the current data source.

However, if you have altered the columns in any way at design time, then the **GridEX** will assume that you wish to maintain the modified grid layout and will not automatically reset the columns.

The **Rebind** method clears all groups and sort keys in a **GridEX** control. If the columns are reset, it clears the formatting conditions too.

Note To force the grid to reset the column bindings even if the columns were modified at design time, invoke the **ClearFields** method immediately before **ReBind**. Conversely, to cancel the grid's automatic layout response and force the grid to use the current column/field layout, invoke the **HoldFields** method immediately before **ReBind**.

Recordset Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a **DAO.Recordset** object defined by **GridEX** control's properties or by an existing **DAO.Recordset** object.

Syntax

object.**Recordset** [= *value*]

The **Recordset** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	An object variable containing a DAO.Recordset object.

Remarks

The **GridEX** control is automatically initialized when the control first appears. If the **DataMode** property is set to **jpgexDAO** and the **DatabaseName**, **RecordSource**, **Exclusive**, **ReadOnly**, **LockType**, **Connect** and **RecordsetType** properties are valid, the control attempts to create a new **Recordset** object based on those properties. This also happens when you set these properties at runtime and use the **Rebind** Method. A clone of this **Recordset** is accessible through the **GridEX** control's **Recordset** property. If, however, one or more of these properties is set incorrectly, an **Error** event is sent to the application when the control attempts to use the properties to create the **Recordset** object.

You can also request the type of **Recordset** to be created by setting the **GridEX** control's **RecordsetType** property. If you don't request a specific type, a Dynaset-type **Recordset** is created. Using the **RecordsetType** property, you can request to create either a Snapshot-, or Table-type **Recordset**. However, if the control can't create the type requested, an **Error** event occurs.

If you create a **Recordset** object using code, you can set the **Recordset** property of the control to this new **Recordset**. Any existing **Recordset** in the control is released and a clone of the **Recordset** assigned replace it.

If you set this property at run time the **DataMode** property of the control is changed to **jpgexDAO**, no matter the previous setting for this property. However, if you try to read the **Recordset** property when the control's mode is other than DAO, a trappable error occurs.

Note You must avoid to use the close event in any variable set to **Recordset** property because it will make invalid this object for any further operations until the **Rebind** method is used.

Data Type

Object

RecordsetType Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating the type of **Recordset** object you want the **GridEX** control to create.

Syntax

object.**RecordsetType** [= *value*]

The **RecordsetType** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A constant or value that specifies a type of Recordset , as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
jgexRSDAOTable	1	A table-type Recordset (DAO mode only)
jgexRSDAODynaset	2	(Default for DAO mode) A dynaset-type Recordset
jgexRSDAOSnapshot	4	A snapshot-type Recordset (DAO mode only)
jgexRSADOKeyset	1	(Default for ADO mode) A Keyset-cursor type.
jgexRSADOStatic	3	A Static-cursor type.

Remarks

If you try to set a recordset type that doesn't correspond to the **DataMode** of the **GridEX** control an error occurs.

For example, if you set the **RecordsetType** property to Static-cursor type and the **DataMode** property is set to DAO an error occurs. However, if you set **RecordsetType** property to Keyset-cursor type while the control's **DataMode** is DAO no error occurs but the control assumes that the **RecordsetType** is table-type.

Note This property has no meaning in unbound mode.

Data Type

jgexRecordsetTypeConstants

RecordSource Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the underlying table, SQL statement, or QueryDef object for the **Recordset** in a **GridEX** control.

Syntax

object.**RecordSource** [= *value*]

The **RecordSource** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression specifying the name of a table, QueryDef or a SQL statement.

Remarks

The **RecordSource** property specifies the source of the recordset in DAO and ADO mode. In unbound mode, this property has no effect.

After changing the value of the **RecordSource** property at run time, you must use the **Rebind** method to enable the change and rebuild the **Recordset**.

Data Type

String

Refresh Method

[See Also](#)

[Example](#)

[Applies To](#)

Forces a complete repaint of a **GridEX** control.

Syntax

object.**Refresh**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Generally, painting is handled automatically while no events are occurring. However, there may be situations where you want the control updated immediately.

The **Refresh** method doesn't reopen the underlying **Recordset** to do so use the **Rebind** method instead.

RefreshGroups Method

[See Also](#)

[Example](#)

[Applies To](#)

Forces a recalculation of groups in a **GridEX** control.

Syntax

object.RefreshGroups allcollapsed

The **RefreshGroups** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>allcollapsed</i>	Optional. A Boolean expression that determines if groups must be collapsed or expanded as described in settings. The default value is False .

Settings

The settings for *allcollapsed* are

Setting	Description
True	The group rows will be collapsed.
False	The group rows will be expanded.

Remarks

If other user changes data that affects the data in the underlying recordset, the actual groups could be changed too. However, the **GridEX** can't know when this happens. In order to force recalculation of groups with the current group settings you may use the **RefreshGroups** method. An example of this is as follows:

```
Private Sub mnuRefreshGroups_Click()  
    If GridEX1.Groups.Count>0 then  
        GridEX1.RefreshGroups  
    End if  
End sub
```

Use the **RefreshGroups** method when there's a reason to believe that other users has changed data in the underlying **Recordset**, making invalid the actual group configuration.

If you use this method in card view, an error occurs.

Note The **RefreshGroups** method also refreshes sort if there is any group in the **GridEX** control.

RefreshSort Method

[See Also](#)

[Example](#)

[Applies To](#)

Forces a **GridEX** control to sort the records.

Syntax

object.**RefreshSort**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

If another user changes data that affects the data in the underlying **Recordset** the record sorting could be lost. However, the **GridEX** can't know when this happens in order to force sorting with the current sort settings you may use the **RefreshSort** method. An example of this is as follows:

```
Private Sub mnuRefreshSort_Click()  
    If GridEX1.SortKeys.Count>0 then  
        GridEX1.RefreshSort  
    End if  
End sub
```

Use the **RefreshGroups** method when there's a reason to believe other users has changed data in the underlying **Recordset** making some of the records be in the wrong order.

Note The **RefreshSort** method doesn't refresh groups in the **GridEX** control

RowBookmark Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value containing a bookmark for a row in the **GridEX** control. Not available at design time

Syntax

object.**RowBookmark**(*rowindex*) [= *value*]

The **RowBookmark** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>rowindex</i>	Required. A long in the range of 1 to the setting of the ItemCount property.
<i>value</i>	A variant expression that represents the bookmark of a row.

Remarks

The **RowBookmark** property is read-only for DAO and ADO modes and is read-write for unbound mode. In unbound mode, you can use this value as a tag property for rows. In bound modes, you can get the **Bookmark** of any row to position the recordset clone of a **GridEX** control as follows:

```
Dim varBookmark as String
DimRowIndex as Long
Dim rstCustomers as recordset
Set rstCustomers = GridEX1.Recordset
RowIndex = GridEX1.RowIndex(GridEX1.Row)
varBookmark = GridEX1.RowBookmark(RowIndex)
rstCustomers.Bookmark = varBookmark
debug.print rstCustomers![CustomerId]
```

Note The *rowindex* parameter in the **RowBookmark** property represents the original position of record and because this position could be changed, when records are sorted or grouped, the *rowindex* doesn't always match with its position.

Data Type

Variant

RowColChange Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when the current cell changes to a different cell.

Syntax

Private Sub *object*_RowColChange([*index* **As Integer**,] **ByVal** *lastrow* **As Long**, **ByVal** *lastcol* **As Integer**)

The **RowColChange** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that uniquely identifies a control if it is in a control array.
<i>lastrow</i>	A long specifying the previous row position.
<i>lastcol</i>	An integer specifying the previous column position.

Remarks

This event occurs whenever the user clicks a cell other than the current cell or when you programmatically change the current cell using the **Col** and **Row** properties. The previous cell position is specified by *lastrow* and *lastcol*. If you edit data and then move the current cell position to a different row, the update events for the original row are completed before another cell becomes the current cell.

RowHeaders Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the row headers are displayed in a **GridEX** control.

Syntax

object.**RowHeaders** [= *value*]

The **RowHeaders** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether row headers are displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	GridEX control's row headers are displayed
False	GridEX control's row headers aren't displayed

Remarks:

When the **GridEX** control is in card view this property has no effect.

Data Type

Boolean

RowHeight Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the height, in twips, of all rows in a **GridEX** control. Read-only.

Syntax

object.**RowHeight**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **GridEX** control automatically calculates the height of rows to accommodate the font and image height.

Data Type

Long

Row Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the current row/card position in a **GridEX** control. Not available at design time.

Syntax

object.Row [= *value*]

The **Row** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A long that represents the row/card position of the current row in a GridEX control.

Remarks:

The **Row** property in **GridEX** represents the position of a record in a **GridEX** control and not its original position in the **Recordset**. If you need to get the original position of a row, you must use the **RowIndex** method as follows:

```
Dim RowPosition as Long
Dim RowIndex as Long
RowPosition = GridEX1.Row
RowIndex = GridEX1.RowIndex(RowPosition)
Debug.print RowPosition,RowIndex
```

The index of a row not always matches with its position, because sorting and/or grouping could change its position.

In some cases, as when a **GridEX** control is grouped the **Row** property can return a value greater than the **ItemCount** setting, this is because the **ItemCount** doesn't count the group rows. If you want to find out the number of rows displayed use the **RowCount** property instead.

Data Type

Long

RowIndex Method

[See Also](#)

[Example](#)

[Applies To](#)

Returns the original index of a row displayed in a **GridEX** control

Syntax

object.RowIndex(rowposition)

The **RowIndex** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>rowposition</i>	A long that identifies the row position whose index is to be retrieved.

Remarks:

In a **GridEX** control, the row position not always matches with its original index so you must differentiate the index of a row and its position. If the **GridEX** control is neither sorted nor grouped, the **RowIndex** will match with its position.

If you use this method in a group row, the return value will be always 0

RowCount Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the count of rows in a GridEX control. Read Only

Syntax

object.**RowCount**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks:

When the records in a **GridEX** control are grouped, the number of rows in a **GridEX** is different to the number of records. For example, if there are four groups and all groups are collapsed, the **RowCount** property will return 4, even when the **ItemCount** property is 100. When these four groups are expanded the **RowCount** will be of 104, that is 100 records and 4 group rows.

Note The **RowCount** property is always equal to the **ItemCount** property if the **GridEX** control is not grouped.

Data Type

Long

SelectionMode Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the style used by a **GridEX** control to highlight a selected cell.

Syntax

object.**SelectionMode** [= *value*]

The **SelectionMode** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A constant or value that specifies the way a GridEX control will highlight a selected cell, as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
jgexEntireRow	0	(Default) When a cell is selected, the entire row is highlighted.
jgexSingleCell	1	When a cell is selected, only that cell is highlighted.

Data Type

Integer

SortKeys Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **SortKeys** object in a **GridEX** control.

Syntax

object.**SortKeys**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **SortKeys** property returns a **SortKeys** collection. You can get a specific **SortKey** through the **Item** property in **SortKeys** object. You can also add and delete **SortKeys**.

Data Type

SortKeys

UnboundAddNew Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs in an unbound **GridEX** control when a new row is added to it. This event alerts your application that it must add a new row of data.

Syntax

Private Sub *object*_**UnboundAddNew**([*index* **As Integer**,] *newrowbookmark* **As Variant**, *varvalues*() **As Variant**)

The **UnboundAddNew** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>Index</i>	An integer that identifies a control if it's in a control array.
<i>newrowbookmark</i>	A variant that represents a bookmark which acts as a unique identifier for each row of data.
<i>varvalues</i>	An array of variants that holds the values entered by the users in the new record.

Remarks

A new row of data can be added, only if the **AllowAddNew** property is **True**.

When the **UnboundAddNew** event is fired the **UnboundUpdate** event will not be triggered.

UnboundDelete Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs whenever a row of data is deleted from the unbound **GridEX**. This event alerts your application that it must delete a row of data.

Syntax

Private Sub *object*_**UnboundDelete**([*index* **As Integer**,] **ByVal** *rowindex* **As Long**, **ByVal** *bookmark* **As Variant**)

The **UnboundDelete** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it's in a control array.
<i>rowindex</i>	A long representing the index of the row to be deleted
<i>bookmark</i>	A value representing the bookmark of the row to be deleted.

Remarks

This event is fired to alert your application that it must delete the row and the event can't be cancelled. However, if you want to cancel the deletion of a particular record you can do it in the **BeforeDelete** event

The *rowindex* parameter is the same as if you get the **RowIndex** of the **Row** property in a **GridEX** control as follows:

```
Private Sub GridEX1_UnboundDelete(ByVal RowIndex as Long, _  
                                   ByVal Bookmark as variant)  
  
    Dim lngIndex as long  
    Dim lngRow as long  
    lngRow = GridEX1.Row  
    lngIndex = GridEX1.RowIndex(lngRow)  
    debug.print lngIndex = RowIndex 'it always print true  
End sub
```

UnboundReadData Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs whenever an unbound **GridEX** control requires data for display, sorting or grouping.

Syntax

```
Private Sub object_UnboundReadData( [ index As Integer, ] ByVal rowindex As Long, ByVal bookmark As Variant, values() As Variant)
```

The **UnboundReadData** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it's in a control array.
<i>rowindex</i>	A long that identifies the original index of the row being retrieved.
<i>bookmark</i>	A variant bookmark which specifies the bookmark set to row using the RowBookmark property
<i>values()</i>	A variant array where the application must set the values corresponding the row being fetched. The bounds of the values parameters starts with 1 to count of columns in a GridEX control.

Remarks

This event retrieves all the column values for any particular row. To this event doesn't matter the row position. The *rowindex* parameter represents the original index of the row. As in rows, the position of columns doesn't matter either. To set the value of third column in the columns collection you may write code as follows:

```
Values(3) = "Column 3"
```

The *bookmark* parameter is provided as a reference. You can not change the bookmark in this event to do so you must use **RowBookmark** property.

The *rowindex* parameter is in the range of 1 to **ItemCount** property value.

UnboundUpdate Event

[See Also](#)

[Example](#)

[Applies To](#)

Occurs when an unbound **GridEX** control has an entire row of modified data to write to the data set. It alerts your application that it must update an edited existing row of data.

Syntax

Private Sub *object*_**UnboundUpdate**([*index* **As Integer**,] **ByVal** *bookmark* **As Variant**, *varvalues*() **As Variant**)

The **UnboundUpdate** event syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that identifies a control if it's in a control array.
<i>bookmark</i>	A bookmark which acts as a unique identifier for each row of data.
<i>varvalues</i>	An array of variants that holds the values entered by the users in the new record.

Remarks

You can't cancel this event to do so you must use the **BeforeUpdate** event.

If the row that is to be updated is a new record, the **UnboundAddNew** event will be fired instead.

If you don't use bookmarks but you need the index of the row to be updated you could write code as follows:

```
DimRowIndex As Long
RowIndex = GridEX1.RowIndex(GridEX1.Row)
Debug.print RowIndex
```

Value Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the value of a column in the current row of a **GridEX** control.

Syntax

object.**Value** (*colindex*) [= *string*]

The **Value** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>colindex</i>	An integer that represent the index of the column.
<i>value</i>	A variant expression specifying the value in the column of the current row.

Data Type

Variant

View Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the way records are displayed in a **GridEX** control.

Syntax

object.**View** [= *value*]

The **View** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the way records are displayed in a GridEX control, as described in Settings.

Settings

The settings for *value* are:

Constant	Setting	Description
jgexTable	0	(Default) (Table view) The records are displayed as a table where each record represents a row.
jgexCard	1	(Card view) The records are displayed as cards where each card represents a record and each row in the card represents a column.

Remarks

If a **GridEX** control is in card view, no matter the setting of the **AllowAddNew** property, the user will not be able to add new records. The user neither will be able to group records as in table view and any attempt of doing it will cause an error.

There are some properties that applies only to a particular view as **CardWidth** properties that only applies to card view or **GroupByBoxVisible** that only applies to table view. If you want to know, whether a particular property applies to a view, search for it in the documentation.

Note In card view any reference to a row must be understood as a card.

Data Type

jgexViewConstants

Column Object

[See Also](#)

[Example](#)

[Properties](#)

Represents a column within a **GridEX** control.

Syntax

gridex.Columns(index)

The **Column** object syntax has these parts:

Part	Description
<i>gridex</i>	An object expression that evaluates to a GridEX control.
<i>index</i>	Either an integer or string that uniquely identifies a member of an object collection. An integer would be the value of the Index property; a string would be the value of the Key property.

Remarks

With a **Column** object, you can modify attributes of the column header as well as the column cells. A **Column** object in the **GridEX** control also represents a field (row) in a card.

To use a **Column** object, you can use the **Columns** property of the **GridEX** control. You can also assign a column to a separate variable dimensioned as a **Column**. The following shows both ways:

```
Dim colTemp as Column
Set colTemp =GridEX1.Columns(1)
colTemp.Caption = "Column 1"
Debug.Print ( colTemp.Caption = GridEX1.Columns(1).Caption )
'Prints True
```

AllowSizing Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether a user can resize a column in the **GridEX** control at run-time.

Syntax

object.AllowSizing [= *value*]

The **AllowSizing** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether a column can be resized, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) User can resize column.
False	User can't resize column.

Remarks

If the **AllowSizing** property is **True**, the mouse pointer turns into a double-headed (Size W E) arrow when positioned over the divider of the specified column, and the user can resize the column by dragging. Any change in column size causes a **ColResize** event.

Data Type

Boolean

Caption Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the text displayed in the column's heading area.

Syntax

object.**Caption** [= *value*]

The **Caption** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that determines what is displayed, as described below.

Remarks

Setting the **Caption** property to an empty string for a **Column** object clears the text in the column's heading area but does not hide the heading. **Column** captions are only displayed if the **GridEX** control's **ColumnHeaders** property is set to **True**.

This **Caption** is also displayed in a card if the column's **ShowCaptionInCardView** property is set to **True**.

Data Type

String

CardCaption Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets whether a column's value will be displayed in the card caption bar while the **GridEX** is in card view.

Syntax

object.**CardCaption** [= *value*]

The **CardCaption** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean that determines if the value of this column is displayed in the card caption. As described in settings.

Settings

The settings for *value* are

Setting	Description
True	The column's value will be displayed in the card caption.
False	The column's value will not be displayed in the card caption.

Remarks

If you set a column's **CardCaption** property to **True**, the column that formerly had this property set to **True** will change its setting to **False**, so only one column can have **CardCaption** property set to **True** at any time.

Even when a column's **CardCaption** property is set to **True**, the column will appear in the card body except if you set its **Visible** property to **False**.

This property has no effect if a **GridEX** control is in table view.

Data Type

Boolean

CardIcon Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets whether a column's icon will be displayed in the card caption while the **GridEX** is in card view.

Syntax

object.**CardIcon** [= *value*]

The **CardIcon** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean that determines if the icon of this column is displayed in the card caption. As described in settings.

Settings

The settings for *value* are

Setting	Description
True	The column's icon will be displayed in the card caption.
False	The column's icon will not be displayed in the card caption.

Remarks

If you set a column's **CardIcon** property to **True**, the column that formerly had this property set to **True**, will change its setting to **False**, so only one column can have **CardIcon** property set to **True** at any time.

Even when a column's **CardIcon** property is set to **True**, the column will appear in the card body except if you set its **Visible** property to false.

This property has no effect if a **GridEX** control is in table view.

Data Type

Boolean

ColPosition Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the position of a column in a **GridEX** control.

Syntax

object.**ColPosition** [= *value*]

The **ColPosition** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	An integer that represents the position of a column in a GridEX control.

Remarks

When a column has changed its position, the column's **Index** property remains the same and is the **ColPosition** property the one that changes. If you want to retrieve a **Column** object by its position, use the **ItemByPosition** method in the **Columns** collection.

Data Type

Integer

ColumnType Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets how column's contents will be displayed.

Syntax

object.**ColumnType** [= *value*]

The **ColumnType** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines how column is displayed, as described in settings.

Settings

The settings for *value* are:

Constant	Value	Description
jgexText	0	(Default) The cells of the column will display only text.
jgexIcon	1	The cells of the column will display only icons
jgexIconAndText	2	The cells of the column will display icons and text.
jgexCheckBox	3	The cells will appear as an embedded check box.

Remarks:

If a column type is icon-type or icon-and-text-type the **GridEX** will display the icon whose **Index** is specified in the **DefaultIcon** property. If the column has a **ValueList**, the **IconIndex** linked to the cell value will be used. Also, if the column's **FetchIcon** property is set to **True**, the **GridImage** whose **Index** was specified in the **FetchIcon** event is used.

Data Type

jgexColumnTypeConstants

DataField Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that represents a field in the underlying recordset for a **GridEX** control.

Syntax

object.**DataField** [= *value*]

The **DataField** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that evaluates to the name of one of the fields in the Recordset object of a GridEX control.

Remarks

If a column's **DataField** property doesn't match with a field in the **Recordset** object no errors occurs, however the column will appear empty and any changes made to that column will not be committed. In unbound mode, this property could be used as a Tag property for the column.

Data Type

String

DefaultIcon Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the index of the **GridImage** displayed in the column's cells.

Syntax

object.DefaultIcon [= *value*]

The **DefaultIcon** syntax has these parts:

Part	Description
<i>object</i>	An objects expression that evaluates to an object in the Applies To list.
<i>value</i>	An integer that refers to the index of the GridImage that will be used as default.

Remarks

Use this property when you want a column that always displays the same icon for all records, instead of using the **FetchIcon** event. If the column's **FetchIcon** property is set to true the icon displayed will be the one retrieved in the **FetchIcon** event for that column.

Note If you try to set this property in a column whose type is text or check box an error occurs.

Data Type

Integer

EditType Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets how column is edited by the user.

Syntax

object.**EditType** [= *value*]

The **EditType** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines how column is edited, as described in settings.

Settings

The settings for *value* are:

Constant	Setting	Description
jgexEditNone	0	(Default) The column is read-only.
jgexEditTextBox	1	When the user tries to edit a column, a Text box appears.
jgexEditCheckBox	2	The column will present a check box for editing.
jgexEditDropDown	3	The column will present a drop down list for editing

Remarks:

Some of the edit types could not be used for all column types. The possible combinations of these two properties are given in the following table

ColumnType	Possible EditType settings
Icon (ColumnType = jgexIcon)	jgexEditNone jgexEditDropDown
Text (ColumnType = jgexText)	jgexEditNone jgexEditTextBox jgexEditDropDown
Icon and text (ColumnType = jgexIconAndText)	jgexEditNone jgexEditTextBox jgexEditDropDown
Check Box (ColumnType = jgexCheckBox)	jgexEditNone jgexEditCheckBox

FetchData Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets whether the **FetchData** event will be fired for a column.

Syntax

object.**FetchData** [= *value*]

The **FetchData** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that controls whether the FetchData event will be fired for that column, as described in settings

Settings:

The settings for *value* are:

Setting	Description
True	The FetchData event will be fired any time the GridEX control needs the value for a cell in the column.
False	(Default) The value of a cell in the column will be retrieved from the Recordset .

Remarks

This property has only effect in DAO and ADO data modes.

Use this property when you want a column that displays a calculated value.

Data Type

Boolean

FetchIcon Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets whether the **FetchIcon** event will be fired for a column.

Syntax

object.**FetchIcon** [= *value*]

The **FetchIcon** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that controls whether the FetchIcon event will be fired for that column, as described in settings

Settings:

The settings for *value* are:

Setting	Description
True	The FetchIcon event will be fired any time the GridEX control needs the GridImage index for a cell in the column.
False	(Default) No FetchIcon event is fired for that column.

Remarks

This property can only be set to **True** if the column's **ColumnType** property is either icon or icon-and-text.

Use this property when you want to display different icons in a column.

Data Type

Boolean

Format Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating the format string for the **Column** object of a **GridEX** control.

Syntax

object.**Format** [= *value*]

The **Format** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that defines how the expression in the Value property is formatted. The default value is a zero-length string ("").

Remarks

See the **Format** function for information about valid format strings.

Data Type

String

GroupEmptyStringCaption Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the string displayed in a group row when the value grouped is an empty string.

Syntax

object.GroupEmptyStringCaption [= *value*]

The **GroupEmptyStringCaption** syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that defines the text that will be displayed in the group row when the value of the group is an empty string or a Null value. The default value is “(none)”

Remarks:

Use this property if you want to display some informational text in a group whose value is an empty string or Null.

Data Type

String

GroupFormat Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating the format string for the value of the group rows when a **GridEX** control is grouped by a column.

Syntax

object.**GroupFormat** [= *value*]

The **GroupFormat** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that defines how the value of the group rows is formatted. The default value is a zero-length string ("").

Remarks

In some cases, you may want to have different formats for a value when it's presented as a cell or as a group. For example, a column with dates may have its **Format** property set to "Medium Date" and its **GroupFormat** property set to "Long Date".

See the **Format** function for information about valid format strings.

Data Type

String

GroupPrefix Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the string displayed in a group row before the group value.

Syntax

object.**GroupPrefix** [= *value*]

The **GroupPrefix** syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that defines the text that will be displayed in a group row before the group value. The default value is a zero-length string ("").

Remarks:

Use this property if you want to display some informational text in a group before its value. For example, if you want to set the **GroupPrefix** property for each column in a **GridEX** control to its **Caption** property plus a ":" you can write code as follows:

```
Dim Col as Column
For each Col in GridEX1.Columns
    Col.GroupPrefix = Col.Caption & ": "
Next
```

Data Type

String

HasValueList Property

[See Also](#)

[Example](#)

[Applies To](#)

Controls whether a column has a list of values in a **GridEX** control.

Syntax

object.**HasValueList** [= *value*]

The **HasValueList** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A boolean expression that control whether a column has a list of values as described in settings

Settings:

The settings for *value* are

Setting	Description
True	The column has a list of values for replacement.
False	(Default) The column hasn't a list of values for replacement.

Remarks:

When you set a column's **HasValueList** property to **False**, the **ValueList** collection for that column is cleared and any reference to its **ValueList** property will return **Nothing**.

Data Type

Boolean

HeaderAlignment, TextAlignment Properties

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that determines the alignment of text in a column of a **GridEX** control.

Syntax

object.**HeaderAlignment** [= *value*]

object.**TextAlignment** [= *value*]

The **HeaderAlignment** and **TextAlignment** property syntax have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that specifies the type of alignment of a column, as described in Settings.

Settings

The settings for *value* are:

Constant	Setting	Description
jgexAlignLeft	0	Text is left aligned.
jgexAlignCenter	1	Text is centered.
jgexAlignRight	2	Text is right aligned.

Data Type

Integer

Index Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns a value that represents the index of an object in a collection. Read only.

Syntax

object.**Index**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks:

The **Index** property is 1 based.

Data Type

Integer

IsGrouped Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns a value that indicates whether a column is grouped. Read only.

Syntax

object.**IsGrouped**

The *object* placeholder represents an object in the Applies To list.

Settings:

The settings for **IsGrouped** property are:

Setting	Description
True	At least one group, in a GridEX control, refers to the column.
False	No groups, in a GridEX control, refer to the column.

Data Type

Boolean

Key Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a string that uniquely identifies a member in a collection.

Syntax

object.**Key** [= *string*]

The **Key** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>string</i>	A unique string identifying a member in a collection.

Remarks

If the string is not unique, an error will occur.

You can set the **Key** property when you use the **Add** method to add an object to a collection.

Data Type

String

ShowCaptionInCardView Property

[See Also](#)

[Example](#)

[Applies To](#)

Controls whether the caption of a column is displayed in the card body.

Syntax

object.**ShowCaptionInCardView** [= *value*]

The **ShowCaptionInCardView** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that controls whether the caption of a column is displayed in a card as described in settings

Settings:

The settings for *value* are

Setting	Description
True	(Default) The caption of the column is displayed at the left of its value in a card.
False	Only the column's value is displayed in a card.

Remarks:

This property has no effect if a **GridEX** control is in table view.

If a column's **ShowCaptionInCardView** is set to **False**, the column's value is displayed beginning at the left of the card.

Data Type

Boolean

SortOrder Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns a value that represents the sort order of an object in a **GridEX** control. Read-Write for **Group** and **SortKey** objects. For **Column** object Read only.

Syntax

object.SortOrder [= *value*]

The **SortOrder** property syntax has these parts

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list
<i>value</i>	A value or constant that represents the sort order of an object as described in settings.

Settings:

The settings for *value* are:

Constant	Setting	Description
jgexSortNone	0	The column is not sorted.
jgexSortAscending	1	The column is sorted in ascending order.
jgexSortDescending	-1	The column is sorted in descending order.

Remarks:

The **Column**'s **SortOrder** property is set when a column is either sorted or grouped.

If you try to set the **Group** or **SortKey** object's **SortOrder** property to **jgexSortNone** an error occurs.

Data Type

jgexSortOrderConstants

SortType Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that indicates the way **GridEX** control sorts values in a column.

Syntax

object.**SortType** [= *value*]

The **SortType** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that represents the way, a GridEX control sorts values in a column, as described in settings

Settings

The settings for *value* are:

Constant	Setting	Description
jgexSortTypeString	1	(Default) Values are sorted as strings.
jgexSortTypeNumeric	2	Values are sorted as numbers. Any non-numeric value is evaluated as 0.
jgexSortTypeDate	3	Values are sorted as dates. Groups ranges are one day.
jgexSortTypeDateTime	4	Values are sorted as dates and times. Groups ranges are one second.
jgexSortTypeTime	5	Values are sorted using the time part of a date and discarding the date. Group ranges are one second.

Data Type

Integer

Tag Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets an expression that stores any extra data needed for your program. Unlike other properties, the value of the **Tag** property isn't used by **GridEX** control; you can use this property to identify objects.

Syntax

object.Tag [= *expression*]

The **Tag** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>expression</i>	A string expression identifying the object. The default is a zero-length string ("").

Remarks

You can use this property to assign an identification string to an object without affecting any of its other property settings or causing side effects.

Data Type

String

ValueList Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **ValueList** collection associated with a column of the **GridEX** control.

Syntax

object.**ValueList**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This property returns nothing if the column's **HasValueList** property is set to **False**.

Data Type

ValueList

Visible Property (Column object)

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether an object is visible or hidden.

Syntax

object.Visible [= *boolean*]

The **Visible** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>boolean</i>	A Boolean expression specifying whether the object is visible or hidden.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Object is visible.
False	Object is hidden.

Data Type

Boolean

Width Property (Column object)

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the width, in twips, of a column.

Syntax

object.**Width** [= *number*]

The **Width** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>number</i>	A Long expression specifying the width, in twips, of an object, as described in Settings.

Remarks

The default value for **Width** is the value of the **DefaultColumnWidth** property of **GridEX**. The **Width** property is given in twips.

Data Type

Long

Columns Collection

[See Also](#)

[Example](#)

[Properties](#)

[Methods](#)

Contains a collection of **Column** objects.

Syntax

gridex.**Columns**

The *gridex* placeholder represents an object expression that evaluates to a **GridEX** control.

Remarks

With the **Columns** collection you can add and remove **Column** objects, count the number of columns, and address individual columns.

The **Columns** collection can be accessed through the **Columns** property of the **GridEX** control.

To get a specific column in the collection you can use the **Item** property or the **ItemByPosition** method.

Add Method (Columns collection)

[See Also](#)

[Example](#)

[Applies To](#)

Adds a **Column** object to the collection and returns a reference to the newly created object.

Syntax

object.**Add** *caption*, *columnType*, *editType*, *key*

The **Add** method syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>caption</i>	Optional. A string expression specifying the Caption for the Column object.
<i>columnType</i>	Optional. A value or constant specifying the ColumnType for the Column object. The available column type are detailed in the ColumnType property (Column object). The default value for this parameter is text-type (ColumnType = jgexText)
<i>editType</i>	Optional. A value or constant specifying the EditType for the Column object. The available edit types are detailed in the EditType property (Column object). The default value for this parameter is textbox-type (EditType = jgexEditTextBox)
<i>key</i>	Optional. A unique string that identifies the Column object. Use this value to retrieve a specific Column object.

Remarks

You can add **Column** objects at design time using the **Columns** tab of the Properties Page of the **GridEX** control. At run time, use the **Add** method to add **Column** objects as in the following code:

```
Dim colColumn as Column
Set colColumn = GridEX1.Columns.Add("name", ,jgexEditNone, "name")
```

Data Type

Column

Clear Method

[See Also](#)

[Example](#)

[Applies To](#)

Removes all objects in a collection.

Syntax

object.**Clear**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

To remove only one object from a collection, use the **Remove** method.

Count Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the number of objects in a collection.

Syntax

object.Count

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Remarks

You can use this property with a **For...Next** statement to carry out an operation on objects in a collection. For example, the following code sets all columns' **ColPosition** property to its original index.

```
For I = 1 To GridEX1.Columns.Count
    GridEX1.Columns(I).ColPosition = I
Next I
```

Data Type

Integer

Item Property (Columns collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a specific **Column** of the collection, either by index or by key.

Syntax

object.**Item**(*index*)

The **Item** property syntax has the following parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>index</i>	Required. An expression that specifies the index of a column in the columns collection. If a numeric expression, index must be a number from 1 to the value of the Count property. If a string expression, index must correspond to the Key property of the column.

Remarks

If the value provided as index doesn't match any existing member of the collection, an error occurs.

Item is the default property for a collection. Therefore, the following lines of code are equivalent:

```
Debug.Print GridEX1.Columns(1).Caption
```

```
Debug.Print GridEX1.Columns.Item(1).Caption
```

Data Type

Column

ItemByPosition Method(Columns collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a specific **Column** based on its position.

Syntax

object.**ItemByPosition**(*position*)

The **ItemByPosition** property syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>position</i>	Required. An expression that specifies the position of a column.

Remarks

When a column changes its position the index property for that column remains the same and its **ColPosition** property is the one that changes. For that reason no matter the times that a column is moved, it always appears in the same position of the collection. However, there are times when you need to get a column by its position, in those cases you need to use this method instead of the **Item** property.

For example, when you get the current column in a **GridEX** control through the **Col** property, the value represents the column's position. If you need to get the **Column** object that represents the current column, you need to get it through the **ItemByPosition** property as follows:

```
Dim colColumn as Column
If GridEX1.Col>0 then
    Set colColumn = GridEX1.Columns.ItemByPosition(GridEX1.Col)
    Debug.Print colColumn.Caption
End if
```

Data Type

Column

Remove Method (Columns collection)

[See Also](#)

[Example](#)

[Applies To](#)

Removes a specific member from the **Columns** collection.

Syntax

object.**Remove** *index*

The **Remove** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer or string that uniquely identifies the column within the collection. Use an integer to specify the value of the Index property; use a string to specify the value of the Key property.

Remarks

To remove all the members of a collection, use the **Clear** method.

SortKey Object

[See Also](#)

[Example](#)

[Properties](#)

Contains information for sorting rows in a **GridEX** control.

Syntax

gridex.SortKeys(index)

The **SortKey** object syntax has these parts:

Part	Description
<i>gridex</i>	An object expression that evaluates to a GridEX control.
<i>index</i>	An integer that represents the value of the Index property.

Remarks

With a **SortKey** object, you can modify the sort settings for a **GridEX** control.

To use a **SortKey** object, you can use the **SortKeys** property of the **GridEX** control. You can also assign a **SortKey** to a separate variable dimensioned as **SortKey**. The following shows both ways:

```
Dim stkTemp as SortKey
Set stkTemp =GridEX1.SortKeys (1)
Debug.Print ( stkTemp Is GridEX1.SortKeys(1) )
'Prints True
```

Group Object

[See Also](#)

[Example](#)

[Properties](#)

Contains information for grouping rows in a **GridEX** control.

Syntax

gridex.Groups (index)

The **Group** object syntax has these parts:

Part	Description
<i>gridex</i>	An object expression that evaluates to a GridEX control.
<i>index</i>	An integer that represents the value of the Index property.

Remarks

With a **Group** object, you can modify the group settings for a **GridEX** control.

To use a **Group** object, you can use the **Groups** property of the **GridEX** control. You can also assign a **Group** to a separate variable dimensioned as **Group**. The following shows both ways:

```
Dim grpTemp as Group
Set grpTemp =GridEX1.Groups(1)
Debug.Print ( grpTemp Is GridEX1.Groups(1))
'Prints True
```

ColIndex Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the index of the column referenced by an object.

Syntax

object.**ColIndex** [= *value*]

The **ColIndex** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	An integer that refers to the index of the column to which the object is attached.

Remarks:

If you set the **ColIndex** property to a value that doesn't represents a **Column** in the **Columns** collection an error occurs.

The change of the **ColIndex** property, in a **Group** or **SortKey** object, forces a recalculation of group rows and/or sort positions of all records.

Data Type

Integer

FmtCondition Object

[See Also](#)

[Example](#)

[Properties](#)

[Methods](#)

Represents the criteria that rows must meet to apply a different format in them.

Syntax

gridex.**FmtConditions** (*index*)

The **FmtCondition** object syntax has these parts:

Part	Description
<i>gridex</i>	An object expression that evaluates to a GridEX control.
<i>index</i>	An integer that represents the value of the Index property.

Remarks

With a **FmtCondition** object, you can have special formatting settings for rows that meet certain criteria.

To use a **FmtCondition** object, you can use the **FmtConditions** property of the **GridEX** control. You can also assign a **FmtCondition** to a separate variable dimensioned as **FmtCondition**. The following shows both ways:

```
Dim ftcTemp as FmtCondition
Set ftcTemp =GridEX1.FmtConditions(1)
Debug.Print ( ftcTemp Is GridEX1.FmtConditions(1))
'Prints True
```

FormatStyle Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **FormatStyle** object attached to a **FmtCondition** object.

Syntax

object.**FormatStyle**

The *object* placeholder represents an object in the Applies To list.

Remarks

Each **FmtCondition** object has a **FormatStyle** object that contains format settings for rows that meet the criteria specified in the object.

Data Type

FormatStyle

Operator Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the operator used for comparison in a **FmtCondition** object.

Syntax

object.Operator [= *value*]

The **Operator** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constants that represents the operator used for comparison as described in settings.

Settings

The settings for *value* are:

Constant	Value	Description
jgexEqual	0	Applies FormatStyle properties to all records where the value of the column attached is equal to the Value1 property.
jgexNotEqual	1	Applies FormatStyle properties to all records where the value of the column attached is different to the Value1 property.
jgexGreaterThan	2	Applies FormatStyle properties to all records where the value of the column attached is greater than the Value1 property.
jgexLessThan	3	Applies FormatStyle properties to all records where the value of the column attached is less than the Value1 property.
jgexGreaterThanOrEqualTo	4	Applies FormatStyle properties to all records where the value of the column attached is greater than or equal to the Value1 property.
jgexLessThanOrEqualTo	5	Applies FormatStyle properties to all records where the value of the column attached is less than or equal to the Value1 property.
jgexBetween	6	Applies FormatStyle properties to all records where the value of the column attached is greater than or equal to the Value1 property and less than or equal to Value2 property.
jgexNotBetween	7	Applies FormatStyle properties to all records where the value of the column attached is less than the Value1 property and greater than Value2 property.

lgexContains	8	Applies FormatStyle properties to all records where the value of the column attached contains Value1 property.
lgexNotContains	9	Applies FormatStyle properties to all records where the value of the column attached doesn't contain Value1 property.

Data Type

lgexConditionOperatorConstants

SetCondition Method

[See Also](#)

[Example](#)

[Applies To](#)

Sets the condition parameters for a **FmtCondition** object.

Syntax

object.**SetCondition** *colindex*, *operator*, *value1*, *value2*

The **SetCondition** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>colindex</i>	Required. An integer expression that represents the index of the column, whose values are to be used in the comparison operation.
<i>operator</i>	Required. A value or constant that specifies the operator used in the comparison operation. The available operators are detailed in the Operator property
<i>value1</i>	Required. A variant that represents the value that is to be compared with the column values.
<i>value2</i>	Optional. A variant that represents the value that is to be compared with the column values.

Value1, Value2 Properties

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the values that are compared with column values.

Syntax:

object.**Value1** [= *value*]

object.**Value2** [= *value*]

The **Value1** and **Value2** property syntax have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A variant expression that represents the value used for comparisons with the attached column values.

Remarks

The **Value2** property setting is used only for between and not between comparisons.

Data Type

Variant

FmtConditions Collection

[See Also](#)

[Example](#)

[Properties](#)

[Methods](#)

Contains a collection of **FmtCondition** objects.

Syntax

gridex.**FmtConditions**

The *gridex* placeholder represents an object expression that evaluates to a **GridEX** control.

Remarks

With the **FmtConditions** collection you can add and remove **FmtCondition** objects, count the number of **FmtCondition** objects, and address individual **FmtCondition** objects.

The **FmtConditions** collection can be accessed through the **FmtConditions** property of the **GridEX** control.

To get a specific **FmtCondition** object in the collection you can use the **Item** property or the **GroupCondition** property.

Add Method (FmtConditions collection)

[See Also](#)

[Example](#)

[Applies To](#)

Adds a **FmtCondition** object to the collection and returns a reference to the newly created object.

Syntax

object.Add colindex, operator, value1, value2, key

The **Add** method syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>colindex</i>	Required. An integer that represents the index of the Column object to attach the FmtCondition .
<i>operator</i>	Optional. A value or constant specifying the Operator property for the FmtCondition object. The available operators are detailed in the Operator property (FmtCondition object). The default value for this parameter is equal operator (Operator = jgexEqual)
<i>value1</i>	Optional. A variant that represents the value that is to be compared with the column values.
<i>value2</i>	Optional. A variant that represents the value that is to be compared with the column values.
<i>Key</i>	Optional. A unique string that identifies the FmtCondition object. Use this value to retrieve a specific FmtCondition object.

Remarks

You can only add **FmtCondition** objects at run time. Use the **Add** method to add **FmtCondition** objects as in the following code:

```
Dim fmcCondition as FmtCondition
Set fmcCondition = GridEX1.FmtConditions.Add(1, jgexEqual, "")
```

Data Type

FmtCondition

ApplyGroupCondition Property

[See Also](#)

[Example](#)

[Applies To](#)

Controls whether the **GroupCondition** property will be valid for a **FmtConditions** collection.

Syntax

object.**ApplyGroupCondition** [= *value*]

The **ApplyGroupCondition** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A boolean expression that control whether the GroupCondition property will be valid for a FmtConditions collection as described in settings.

Settings

The settings for *value* are:

Setting	Description
True	The GroupCondition property is valid and it will be applied whenever a GridEX control is grouped.
False	(Default) No conditional formatting is applied in the group rows.

Remarks:

When you need to highlight records that meet certain criteria and you want this highlight to be expanded through the groups you must use the **GroupCondition** property. In order to be able to use it you first must set the **ApplyGroupCondition** property to **True**.

Data Type

Boolean

GroupCondition Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns a **FmtCondition** object that is applied in the group rows in a **GridEX** control.

Syntax

object.**GroupCondition**

The *object* placeholder represents an object in the Applies To list.

Remarks:

The **GroupCondition** property returns a **FmtCondition** object that is applied to the group rows when one or more records in the group meet the criteria specified by the **FmtCondition**'s properties. The **FormatStyle** properties are applied only to the group rows and if you want to apply it to all the rows, you must add a **FmtCondition** to the **FmtConditions** collection with the same property settings.

Data Type

FmtCondition

GroupConditionCountTitle Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the text displayed in a group row when one or more rows in the group meet the criteria specified in the **GroupCondition** object's properties.

Syntax

object.**GroupConditionCountTitle** [= *value*]

The **GroupConditionCountTitle** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that represents the text displayed with count of rows that meet the criteria specified in the GroupCondition property settings. The default value is "items"

Remarks:

Whenever a group row has records that meet the criteria specified in the **GroupCondition** property settings, the group row is displayed with the **GroupCondition**'s **FormatStyle** property settings. If the **ShowGroupConditionCount** is set to **True**, the group row will display the row count that meet the criteria followed by the **GroupConditionCountTitle** property setting and enclosed by parenthesis.

This property has no use if the **ApplyGroupCondition** or **ShowGroupConditionCount** properties are set to **False**.

Data Type

String

Item Property (FmtConditions collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a specific **FmtCondition** of the **FmtConditions** collection either by index or by key.

Syntax

object.**Item**(*index*)

The **Item** property syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>index</i>	Required. An expression that specifies the position of a member of the collection. If a numeric expression, index must be a number from 1 to the value of the Count property. If a string expression, index must correspond to the Key property of the member.

Remarks

If the value provided as *index* doesn't match any existing member of the collection, an error occurs.

Item is the default property for a collection. Therefore, the following lines of code are equivalent:

```
Debug.Print GridEX1.FmtConditions(1).ColIndex
```

```
Debug.Print GridEX1.FmtConditions.Item(1).ColIndex
```

Data Type

FmtCondition

Remove Method (FmtConditions collection)

[See Also](#)

[Example](#)

[Applies To](#)

Removes a specific member from the **FmtConditions** collection.

Syntax

object.Remove index

The **Remove** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer or string that uniquely identifies the FmtCondition within the collection. Use an integer to specify the value of the Index property; use a string to specify the value of the Key property.

Remarks

To remove all the members of a collection, use the **Clear** method.

ShowGroupConditionCount Property

[See Also](#)

[Example](#)

[Applies To](#)

Controls whether the count of rows, that meet the criteria in **GroupCondition** property settings, will be displayed in a group row.

Syntax

object.**ShowGroupConditionCount** [= *value*]

The **ShowGroupConditionCount** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that controls whether the count of rows, that meet criteria in GroupCondition property settings, is displayed in the group row as described in settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) The group row displays the row count.
False	The group row doesn't display the row count.

Data Type

Boolean

FormatStyle Object

[See Also](#)

[Example](#)

[Properties](#)

Contains the format settings to apply, in rows that meet criteria, in its **FmtCondition** parent.

Syntax

object.**FormatStyle**

The *object* placeholder represents an object expression that evaluates to a **FmtCondition** object.

Remarks

With a **FormatStyle** object, you can specify the format settings for rows that meet criteria in the **FmtCondition** object parent.

To use a **FormatStyle** object, you can use the **FormatStyle** property of a **FmtCondition** object. You can also assign a **FormatStyle** to a separate variable dimensioned as **FormatStyle**. The following shows both ways:

```
Dim ftsTemp as FormatStyle
Set ftsTemp = GridEX1.FmtConditions(1).FormatStyle
Debug.Print (ftsTemp Is GridEX1.FmtConditions(1).FormatStyle)
'Prints True
```

ForeColor Property (FormatStyle object)

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the foreground color of the **FormatStyle** object.

Syntax

object.**ForeColor** [= *color*]

The **ForeColor** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A numeric expression that specifies the color.

Remarks

The **ForeColor** property refers to the text color used to display the rows that meet criteria specified in the **FmtCondition** settings

Data Type

OLE_COLOR

FontBold, FontItalic, FontStrikeThru, FontUnderline Properties

[See Also](#)

[Example](#)

[Applies To](#)

Return or set font styles in the following formats: Bold, Italic, Strikethrough, and Underline.

Syntax

object.**FontBold** [= *boolean*]

object.**FontItalic** [= *boolean*]

object.**FontStrikeThru** [= *boolean*]

object.**FontUnderline** [= *boolean*]

The **FontBold**, **FontItalic**, **FontStrikeThru**, and **FontUnderline** property syntax have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>boolean</i>	A Boolean expression specifying the font style as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Turns on the formatting in that style.
False	(Default) Turns off the formatting in that style.

Remarks

Use these font properties to format text in rows that meet criteria specified in **FmtCondition** object settings.

Data Type

Boolean

GridImage Object

[See Also](#)

[Example](#)

[Properties](#)

Represents an image used in a **GridEX** control.

Syntax

gridex.**GridImages** (*index*)

The **GridImage** object syntax has these parts:

Part	Description
<i>gridex</i>	An object expression that evaluates to a GridEX control.
<i>index</i>	An integer that represents the value of the Index property.

Remarks

A **GridImage** object is used in a **GridEX** control to draw icons in cells.

To use a **GridImage** object, you can use the **GridImages** property of the **GridEX** control. You can also assign a **GridImage** to a separate variable dimensioned as **GridImage**. The following shows both ways:

```
Dim gimTemp as GridImage
Set gimTemp =GridEX1.GridImages(1)
Debug.Print ( gimTemp Is GridEX1.GridImages(1))
'Prints True
```

Picture Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a graphic to be displayed.

Syntax

object.**Picture** [= *picture*]

The **Picture** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>picture</i>	A Picture object containing an icon or bitmap.

Remarks

The **Picture** dimensions remains unchanged even when the **GridImages** object resizes the picture to accommodate the graphic to the **ImageWidth** and **ImageHeight** properties.

Data Type

StdPicture

GridImages Collection

[See Also](#)

[Example](#)

[Properties](#)

[Methods](#)

Contains a collection of **GridImage** objects.

Syntax

gridex.**GridImages**

The *gridex* placeholder represents an object expression that evaluates to a **GridEX** control.

Remarks

With the **GridImages** collection you can add and remove **GridImage** objects, count the number of **GridImage** objects, and address individual **GridImage** objects.

The **GridImages** collection can be accessed through the **GridImages** property of the **GridEX** control.

To get a specific **GridImage** object in the collection you can use the **Item** property.

Add Method (GridImages collection)

[See Also](#)

[Example](#)

[Applies To](#)

Adds a **GridImage** object to the collection and returns a reference to the newly created object.

Syntax

object.**Add** *picture*

The **Add** method syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to a Buttons collection.
<i>picture</i>	Required. Specifies the Picture to be added to the collection

Remarks

The **GridImages** collection is a 1-based collection.

You can load either bitmaps or icons into a **GridImage** object.

Data Type

GridImage

Item Property (GridImages collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a specific **GridImage** of the **GridImages** Collection.

Syntax

object.**Item**(*index*)

The **Item** property syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>index</i>	Required. An integer that specifies the index of a GridImage in the GridImages collection.

Remarks

If the value provided as index doesn't match any existing member of the collection, an error occurs.

Item is the default property for a collection. Therefore, the following lines of code are equivalent:

```
Debug.Print GridEX1.GridImages(1).Index
```

```
Debug.Print GridEX1.GridImages.Item(1).Index
```

Data Type

GridImage

Remove Method (GridImages collection)

[See Also](#)

[Example](#)

[Applies To](#)

Removes a specific member from the **GridImages** collection.

Syntax

object.**Remove** *index*

The **Remove** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that represents the index of a GridImage within the GridImages collection.

Remarks

To remove all the members of a collection, use the **Clear** method.

Groups Collection

[See Also](#)

[Example](#)

[Properties](#)

[Methods](#)

Contains a collection of **Group** objects.

Syntax

gridex.**Groups**

The *gridex* placeholder represents an object expression that evaluates to a **GridEX** control.

Remarks

With the **Groups** collection you can add and remove **Group** objects, count the number of **Group** objects, and address individual **Group** objects.

The **Groups** collection can be accessed through the **Groups** property of the **GridEX** control.

To get a specific **Group** object in the collection you can use the **Item** property.

Note The **Groups** collection can have only four items at any time. If you try to add more than four items an error occurs.

Add Method (Groups Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Adds a **Group** object to the collection and returns a reference to the newly created object.

Syntax

object.Add colindex, sortorder, index

The **Add** method syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>colindex</i>	Required. An integer that represents the index of the column whose values will be grouped.
<i>sortorder</i>	Required. A value or constant that specifies the sort order of an object. The available sort orders are detailed in the SortOrder property.
<i>index</i>	Optional. An integer that specifies the index in which the group will be added.

Remarks

The **Groups** collection is a 1-based collection.

Data Type

Group

Item Property (Groups Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a specific **Group** of the **Groups** Collection.

Syntax

object.**Item**(*index*)

The **Item** property syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>index</i>	Required. An integer that specifies the position of a Group in the Groups collection.

Remarks

If the value provided as index doesn't match any existing member of the collection, an error occurs.

Item is the default property for a collection. Therefore, the following lines of code are equivalent:

```
Debug.Print GridEX1.Groups(1).ColIndex
```

```
Debug.Print GridEX1.Groups.Item(1).ColIndex
```

Data Type

Group

Remove Method (Groups Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Removes a specific member from the **Groups** collection.

Syntax

object.**Remove** *index*

The **Remove** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that represents the index of a Group within the Groups collection.

Remarks

To remove all the members of a collection, use the **Clear** method.

SortKeys Collection

[See Also](#)

[Example](#)

[Properties](#)

[Methods](#)

Contains a collection of **SortKey** objects.

Syntax

gridex.**SortKeys**

The *gridex* placeholder represents an object expression that evaluates to a **GridEX** control.

Remarks

With the **SortKeys** collection you can add and remove **SortKey** objects, count the number of **SortKey** objects, and address individual **SortKey** objects.

The **SortKeys** collection can be accessed through the **SortKeys** property of the **GridEX** control.

To get a specific **SortKey** object in the collection you can use the **Item** property.

Note The **SortKeys** collection can have only four items at any time. If you try to add more than four items an error occurs.

Add Method (SortKeys Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Adds a **SortKey** object to a **SortKeys** collection and returns a reference to the newly created object.

Syntax

object.Add colindex, sortorder, index

The **Add** method syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>colindex</i>	Required. An integer that represents the index of the column whose values will be grouped.
<i>sortorder</i>	Required. A value or constant that specifies the sort order of an object. The available sort orders are detailed in the SortOrder Property.
<i>index</i>	Optional. An integer that specifies the index in which the SortKey will be added.

Remarks

The **SortKeys** collection is a 1-based collection.

Data Type

SortKey

Item Property (SortKeys Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a specific **SortKey** of the **SortKeys** Collection.

Syntax

object.**Item**(*index*)

The **Item** property syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>index</i>	Required. An integer that specifies the position of a SortKey in the SortKeys collection.

Remarks

If the value provided as index doesn't match any existing member of the collection, an error occurs.

Item is the default property for a collection. Therefore, the following lines of code are equivalent:

```
Debug.Print GridEX1.SortKeys(1).ColIndex
```

```
Debug.Print GridEX1.SortKeys.Item(1).ColIndex
```

Data Type

SortKey

Remove Method (SortKeys Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Removes a specific member from the **SortKeys** collection.

Syntax

object.Remove index

The **Remove** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that represents the index of a SortKey within the SortKeys collection.

Remarks

To remove all the members of a collection, use the **Clear** method.

ValueItem Object

[See Also](#)

[Example](#)

[Properties](#)

Represents a text for replacement when a cell matches its value.

Syntax

object.Item(*index*)

The **ValueItem** object syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	An integer that represents the value of the Index property.

Remarks

With a **ValueItem** object, you can associate a text and/or and image to given value.

To use a **ValueItem** object, you can use the **ValueItem** property for a column in a **GridEX** control. You can also assign a **ValueItem** to a separate variable dimensioned as **ValueItem**. The following shows both ways:

```
Dim vitTemp as ValueItem
Set vitTemp = GridEX1.Columns(1).ValueList(1)
Debug.Print ( vitTemp Is GridEX1.Columns(1).ValueList(1) )
'Prints True
```

IconIndex Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the index of the **GridImage** that corresponds to the value in a **ValueItem** object.

Syntax

object.**IconIndex** [= *value*]

The **IconIndex** property syntax has these parts:

Part	Description
<i>object</i>	An objet expression that evaluates to an object in the Applies To list.
<i>value</i>	An integer that represents the index of a GridImage in the GridImages collection.

Remarks:

If you set the **IconIndex** property to 0, no image is used.

Data Type

Integer

Text Property (ValueItem object)

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the text that corresponds to the value in a **ValueItem** object.

Syntax

object.**Text** [= *value*]

The **Text** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A string expression that will be used to replace a the cell contents when the cell value match with the Value property setting in a ValueItem object

Data Type

String

Value Property (ValueItem object)

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the value of a **ValueItem** object.

Syntax

object.**Value** [= *value*]

The **Value** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A variant expression that represents the value of a ValueItem object.

Remarks:

The **Value** property setting must be unique in a **ValueList** collection. Any attempt to set the **Value** property to a value that is already in the list will cause an error.

Data Type

Variant

ValueList Collection

[See Also](#)

[Example](#)

[Properties](#)

[Methods](#)

Represents a list of **ValueItem** objects.

Syntax

column.**ValueList**

The *column* placeholder represents an object expression that evaluates to a **Column** object.

Remarks

With the **ValueList** collection you can add and remove **ValueItem** objects, count the number of **ValueItem** objects, and address individual **ValueItem** objects.

The **ValueList** collection can be accessed through the **ValueList** property of a **Column** object.

To get a specific **ValueItem** object in the collection you can use the **Item** or the **ItemByValue** properties.

Add Method (ValueList Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Adds a **ValueItem** object to a **ValueList** collection and returns a reference to the newly created object.

Syntax

object.**Add** *value*, *text*, *iconindex*

The **Add** method syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>value</i>	Required. A variant expression that represents the unique value for the ValueItem object.
<i>text</i>	Optional. A string expression that represents the text used to replace the value when is displayed.
<i>iconindex</i>	Optional. An integer that specifies the index of a GridImage in the GridImages collection.

Remarks

The **ValueList** collection is a 1-based collection.

Data Type

ValueItem

Item Property (ValueList Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a specific **ValueItem** of the **ValueList** collection.

Syntax

object.**Item**(*index*)

The **Item** property syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>index</i>	Required. A long that specifies the position of a ValueItem in the ValueList collection.

Remarks

If the value provided as index doesn't match any existing member of the collection, an error occurs.

Item is the default property. Therefore, the following lines of code are equivalent:

```
Debug.Print GridEX1.Columns(1).ValueList(1).Text
```

```
Debug.Print GridEX1Columns.item(1).ValueList.Item(1).Text
```

Data Type

ValueItem

ItemByValue Method (ValueList Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Returns a **ValueItem** given its value.

Syntax

object.**ItemByValue**(*value*)

The **ItemByValue** method syntax has these parts:

Part	Description
<i>object</i>	Required. An object expression that evaluates to an object in the Applies To list.
<i>value</i>	Required. A variant that specifies the unique value of a ValueItem in the ValueList collection.

Remarks

If the *value* parameter doesn't match any existing member of the collection, an error occurs.

Data Type

ValueItem

Remove Method (ValueList Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Removes a specific **ValueItem** from the **ValueList** collection.

Syntax

object.**Remove** *index*

The **Remove** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>index</i>	A long that represents the Index property of a ValueItem object.

Remarks

To remove all the members use the **Clear** method.

RemoveByValue Method (ValueList Collection)

[See Also](#)

[Example](#)

[Applies To](#)

Removes a specific **ValueItem** from the **ValueList**.

Syntax

object.**RemoveByValue** *value*

The **RemoveByValue** method syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A variant that represents the Value of a ValueItem object.

Remarks

To remove all the members, use the **Clear** method.

ImageHeight, ImageWidth Properties

[See Also](#)

[Example](#)

[Applies To](#)

- The **ImageHeight** property returns or sets the height of **GridImage** objects in a **GridEX** control.
- The **ImageWidth** property returns or sets the width of **GridImage** objects in a **GridEX** control.

Syntax

object.**ImageHeight** [= *value*]

object.**ImageWidth** [= *value*]

The **ImageHeight** and **ImageWidth** property syntax have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Long expression that specifies the dimensions, in pixels, of GridImage objects in a GridEX control.

Remarks

Both height and width are measured in pixels. All **GridImage** objects in a **GridImages** collection have the same height and width.

When a **GridImages** contains no **GridImage** objects, you can set both **ImageHeight** and **ImageWidth** properties. All images added to the **GridImages** collection are stretched, in case they needed, to accommodate the dimensions specified in **ImageHeight** and **ImageWidth** properties.

Data Type

Long

HeaderIcon Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the index of the **GridImage** displayed in the column's header.

Syntax

object.HeaderIcon [= *value*]

The **HeaderIcon** syntax has these parts:

Part	Description
<i>object</i>	An objects expression that evaluates to an object in the Applies To list.
<i>value</i>	An integer that refers to the index of the GridImage that will be used in the column header.

Data Type

Integer

ExpandAll Method

[See Also](#)

[Example](#)

[Applies To](#)

Expands all group rows in **GridEX** control.

Syntax

object.**ExpandAll**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This method has no use if the **GridEX** control isn't grouped.

CollapseAll Method

[See Also](#)

[Example](#)

[Applies To](#)

Collapses all group rows in a **GridEX** control.

Syntax

object.**CollapseAll**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This method has no use if the **GridEX** control isn't grouped.

ColFromPoint Method

[See Also](#)

[Example](#)

[Applies To](#)

Returns the **Column** object that contains the coordinates provided as argument.

Syntax

object.**ColFromPoint** *x, y*

The **ColFromPoint** method syntax has these parts:

Part	Description
<i>object</i>	An objects expression that evaluates to an object in the Applies To list.
<i>x,y</i>	Coordinates of a target column.

Remarks

If the point specified in the *x* and *y* parameters is not contained in any column the return value is **Nothing**.

Data Type

Column

HitTest Method

[See Also](#)

[Example](#)

[Applies To](#)

Returns a value that represents the part of a **GridEX** control that contains the point specified.

Syntax

object.**HitTest** *x, y*

The **HitTest** method syntax has these parts:

Part	Description
<i>object</i>	An objects expression that evaluates to an object in the Applies To list.
<i>x,y</i>	Coordinates to search.

The settings for the *return value* are:

Constant	Setting	Description
jgexHTNoWhere	0	The point is outside the client area of the GridEX control.
jgexHTGroupByBox	1	The point is in the group by box.
jgexHTColumnHeader	2	The point is in the column header.
jgexHTRowHeader	3	The point is in the row header.
jgexHTNewRow	4	The point is in the top new row.
jgexHTCell	5	The point is in a cell.
jgexHTCard	6	The point is in card.
jgexHTBackGround	7	The point is in the background.

Remarks

The **HitTest** method provides a way to determine in which part a **GridEX** control is a given point. If you want a more detailed information, you can use the **ColFromPoint** and the **RowFromPoint** methods also.

Data Type

jgexHitTestConstants

LockType Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets the type of locking for the **Recordset**.

Syntax

object.**LockType** [= *value*]

The **LockType** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the lock type of the recordset created in a GridEX control, as described in Settings.

Settings

The settings for *value* are:

Constant	Setting	Description
jgexLockReadOnly	1	Prevents user from making changes to the recordset
jgexLockPessimistic	2	Uses pessimistic locking to determine how changes are made to the Recordset in a multi-user environment.
jgexLockOptimistic	3	Uses optimistic locking to determine how changes are made to the Recordset in a multi-user environment.
jgexLockBatchOptimistic	4	Enables batch optimistic updates (ADO mode only)

Remarks:

This property is only used in **DAO** and **ADO** modes to create the **Recordset**. For further information about locking, see **DAO** and/or **ADO** documentation.

Data Type

jgexLockTypeConstants

MoveFirst Method

[See Also](#)

[Example](#)

[Applies To](#)

Sets the first record open in a **GridEX** control as the current row.

Syntax

object.**MoveFirst**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

If a **GridEX** control is grouped, only the open rows that represent records in the **Recordset** are considered.

MoveLast Method

[See Also](#)

[Example](#)

[Applies To](#)

Sets the last record open in a **GridEX** control as the current row.

Syntax

object.**MoveLast**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

If a **GridEX** control is grouped, only the open rows that represent records in the **Recordset** are considered.

MoveNext Method

[See Also](#)

[Example](#)

[Applies To](#)

Moves to the next open record in a **GridEX** control.

Syntax

object.**MoveNext**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

If a **GridEX** control is grouped, only the open rows that represent a record in the **Recordset** are considered.

MovePrevious Method

[See Also](#)

[Example](#)

[Applies To](#)

Moves to the previous open record in a **GridEX** control.

Syntax

object.**MovePrevious**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

If a **GridEX** control is grouped, only the open rows that represent a record in the **Recordset** are considered.

MoveRelative Method

[See Also](#)

[Example](#)

[Applies To](#)

Moves the current row to the position specified.

Syntax

object.**MoveRelative** *nRows*

The **MoveRelative** method syntax has these parts:

Part	Description
<i>object</i>	An objects expression that evaluates to an object in the Applies To list.
<i>nRows</i>	A long expression specifying the number of rows the position will move. If <i>nRows</i> is greater than 0, the position is moved forward (toward the last row). If <i>nRows</i> is less than 0, the position is moved backward (toward the first row).

Remarks

If a **GridEX** control is grouped, only the open rows that represent a record in the **Recordset** are considered.

MoveToBookmark Method

[See Also](#)

[Example](#)

[Applies To](#)

Sets the current row in a **GridEX** control to the one that matches the Bookmark.

Syntax

object.**MoveToBookmark** *vBookmark*

The **MoveToBookmark** method syntax has these parts:

Part	Description
<i>object</i>	An objects expression that evaluates to an object in the Applies To list.
<i>vBookmark</i>	A Variant value identifying a bookmark.

Remarks

If the row with the bookmark specified is hidden because its group is closed, the **GridEX** control will open the necessary row groups in order to ensure the visibility of the row.

Options Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value that specifies one or more characteristics of the **Recordset** object in the **GridEX** control.

Syntax

object.**Options** [= *value*]

The **Options** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A long value that specifies the characteristics of the Recordset .

Remarks:

For more information about the options that could be used see DAO and/or ADO documentation.

Data Type

Long

ReadOnly Property

[See Also](#)

[Example](#)

[Applies To](#)

Returns or sets a value indicating whether the **Database** object, in a **GridEX** control, must be opened as read only.

Syntax

object.**ReadOnly** [= *value*]

The **ReadOnly** property syntax has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that determines whether the underlying Database object is opened as read only, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The Database object underlying the GridEX control is opened as read only.
False	(Default) The Database object underlying the GridEX control is opened as read write.

Remarks

This property is used in DAO mode only. The **ReadOnly** property is used to open the **Database** object that holds the **Recordset**.

Data Type

Boolean

RowFromPoint Method

[See Also](#)

[Example](#)

[Applies To](#)

Returns the row that contains the given point.

Syntax

object.**RowFromPoint** *x, y*

The **RowFromPoint** method syntax has these parts:

Part	Description
<i>object</i>	An objects expression that evaluates to an object in the Applies To list.
<i>x,y</i>	Coordinates of a target row.

Remarks

If the point specified in the *x* and *y* parameters is not contained in any row the return value is 0.

Data Type

Long

Update Method

[See Also](#)

[Example](#)

[Applies To](#)

Commit changes made to the current row writing to the database and re-position the record in case that needed.

Syntax

object.**Update**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The update method also searches if the current row position is still valid. If it isn't and the **AutomaticArrange** property is set to **True**, the **GridEX** control will reposition the row. This could be necessary when changes to the record where made for other user or with the **Recordset** clone.

Delete Method

[See Also](#)

[Example](#)

[Applies To](#)

Deletes the current row in a **GridEX** control.

Syntax

object.**Delete**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Using the **Delete** method is the same as pressing the DEL key while the entire row is selected.

SearchNewRecords Method

[See Also](#)

[Example](#)

[Applies To](#)

Searches for records that could be added after a group or sort operation.

Syntax

object.**SearchNewRecords**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Use this method when you add records to **Recordset** clone of a **GridEX** control, and want that those records be displayed in the control.

