

TrueGrid



A Data-Aware Grid for Visual Basic 3.0 Data Access

This is a demo version of TrueGrid. We believe we've created the easiest and most powerful grid available and appreciate your interest in this demo. It is identical in features and functionality to the full TrueGrid product, except that it will only operate in Visual Basic design mode. We want to make it easy for you to get a good idea of what TrueGrid offers, how it operates, and how flexible it is.

We've put together a demo program which runs during design-time. The demo program is called TGDEMO.MAK. To run the demo, you should:






1. Run Visual Basic.
2. Open the TGDEMO.MAK project.
3. Run the application.

The TGDEMO program is self-explanatory and has a window called the Navigator which describes each of the demos and how to operate them. Running TGDEMO is the quickest way to understand how TrueGrid operates, but you can also develop your own design-time applications using the demo. We've provided documentation with the demo, so you can understand how the grid operates.

If you like TrueGrid, and want one for your very own, you can order one directly from Apex at 800-858-APEX. You can also fax in your order at 412-681-4384. For other questions and information, call Apex at 412-681-4343. We would be glad to provide you with some assistance with the demo, but can't provide the level of support we offer for product owners. You can contact us via CompuServe (Brian Hess 71053,1062) or Internet (truegrid@apexsc.com).

If you want, you can begin developing using the demo version of TrueGrid, since it has the same product features as the production version. However, when you purchase a production TrueGrid, you will need to recreate any TrueGrid controls you were using in the demo and reset the design-time properties they way they were originally.

Here's the table of contents for the on-line TrueGrid documentation:

-  Overview of TrueGrid
-  TrueGrid Pro Features
-  TrueGrid Properties and Events, Grouped by Functional Area
-  Layout Editor Reference
-  Contacting Apex

This is the "Gunsmoke" Version (2.1) of TrueGrid.

Copyright (c) Apex Software Corporation, 1994. All rights reserved.

Contacting Apex

If you have any problems using TrueGrid, or encounter problems which you believe may be software bugs, please contact us and we will do our best to assist you in solving your problem. For technical questions and support, *please* use CompuServe if possible. We can respond more quickly and effectively on CompuServe. If you don't have a CompuServe account, please use fax or telephone as listed below.

Orders: [800-858-APEX](tel:800-858-APEX)

CompuServe: [71053,1062](tel:710531062) (Tech Support)

By phone: [412-681-4738](tel:412-681-4738)

By mail: Apex Software Corporation
4516 Henry Street
Pittsburgh, PA 15213

By fax: [412-681-4384](tel:412-681-4384)

Our office hours are 9 a.m. - 6 p.m. (EST)



Overview

TrueGrid is a bound database grid for Visual Basic 3.0. Using technology derived from our Agility database product, TrueGrid manages the interface with the database completely, freeing you to concentrate on important application-specific tasks.

TrueGrid is very easy to use, requiring no more work than simply dropping the grid on a form and setting the DataSource property--that's all! You don't need to write a single line of code to get the grid up and running, making it ideal for an instant data browser or data display window.

This new version of TrueGrid also adds significant power to the original design. Full Excel-like split capability, customizable fonts and colors, calculated and unbound columns, and drag-and-drop features are just some of the things which have been added. Yet, TrueGrid is still very easy to use, and the new features are there only if you want to use them.

The grid works with all formats that Visual Basic recognizes. Any database that you can use with Visual Basic you can use with the grid. See [Summary of Features](#) for an overview of some of TrueGrid's features.

We've spent a lot of time trying to make TrueGrid *the* inexpensive alternative for Visual Basic data access users. We hope you enjoy using it. If you need [technical assistance](#), please call and talk to us. We'd be glad to help.

The How to Use TrueGrid (not included) section provides detailed examples and instructions on how to use the grid. If you're already very familiar with Visual Basic and other custom controls, you may want to quickly peruse the [TrueGrid Properties and Events](#) section.



TrueGrid Pro Features

The following is a summary of the major features of TrueGrid Pro:

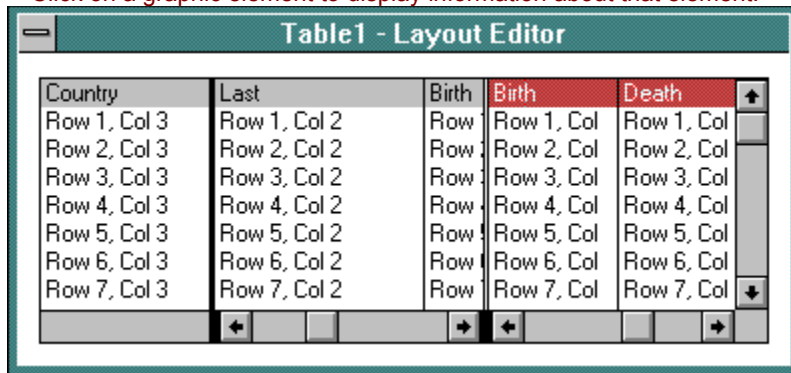
- 1. Easy To Use:** TrueGrid's features are familiar, intuitive, and easy to use. In most cases, they require no code.
- 2. Drop and Go:** Simply drop TrueGrid on a Visual Basic form, set the DataSource property, and instantly you have a fully functional database-aware browse table - without writing a single line of code!
- 3. Database Formats:** Supports all database formats Visual Basic recognizes.
- 4. No Size Limit:** Automatically handles databases of any size - programmers do not need to adjust any virtual memory settings. There is no delay or additional memory overhead for large databases.
- 5. Fast and Efficient:** Only data required for display will be retrieved, TrueGrid is simply the fastest grid in the market.
- 6. Bound Controls:** Automatically works with other Visual Basic bound controls. All Data Control operations, data validation events, and error trapping are supported seamlessly.
- 7. Editing and Data Validation:** In-cell editing with automatic update and data validation events.
- 8. DropDown Text Edit:** A multi-line text control will dropdown automatically when editing large data.
- 9. Configurability:** Most properties are configurable at both design time and by code. In addition, splits, column width and column order are configurable interactively by end-users at run time.
- 10. Layout Editor:** Allows programmers to visually and interactively configure various column and split properties at design time. Features such as locked columns are automated. TrueGrid is the only grid that allows the programmer to visually configure the appearance of the grid at design time.
- 11. Tab and Arrow Keys:** Complete control of the use of the tab and arrow keys to navigate through the cells of the grid or to move focus to another control.
- 12. AddNew:** Allow end-users to enter a new record at the bottom of the grid with automatic update and error trapping.
- 13. Select and Mark:** Select and mark non-contiguous records by mouse clicks, or by code.
- 14. Callback:** TrueGrid can be used with data managed by programmers; it doesn't have to be bound to a Data Control.
- 15. EditMask:** The EditMask property fully supports the Visual Basic Format\$ string.
- 16. Cross-Form Binding:** TrueGrid Pro can be bound to a Visual Basic Data Control in another form.
- 17. Splits:** Splits a grid into any number of independent grids. The splits can scroll simultaneously or independently, or be locked from scrolling. An Excel-like split bar allows end-users to create splits interactively. Programmer can decide how much interactive control the end-users have.
- 18. Per-Cell Color and Font:** Control the color and font style of each individual cell based on cell contents and cell status. If cell contents or status are updated, the color and font will be adjusted automatically.
- 19. Indentation:** Control per-cell text indentation, ideal for building outlines and subtitles in a grid column.
- 20. Multi-Line Row and Heading:** Heading and row heights are independently controlled.
- 21. Drop Down Combo or List Box:** Easy to program features to drop down a combo or list box, or

even any Visual Basic control from a cell.

- 22. Drag and Drop:** Easy to program features to allow drag and drop of data to and from other controls.
- 23. Calculated Columns:** Create columns to display calculated expressions simply by entering a Visual Basic expression in a ColumnExpression property. No code required. Visual Basic constants (e.g., null, True, False, date constants, etc.), operators, field names, functions (over 35 Visual Basic functions are supported), and even other calculated columns are supported. TrueGrid automatically resolves all dependencies and calculation orders.
- 24. In-Cell Graphic:** You can display graphical images in a cell, not just textual information.
- 25. In-Cell Combo Box and Radio-Button:** For easy data entry and inspection; eliminate end-user typing and error.
- 26. Value List:** Values from bound or calculated columns can be translated to any textual strings or pictures for display purposes. For example, (0, 1) may be displayed as (Yes, No), country names may be displayed as bitmaps of their respective flags. Value lists can be defined at design time using the Layout Editor - no code necessary!
- 27. Column Summaries:** Built-in properties to support column summaries such as column sum. An event will be fired when the column sum changes.
- 28. Built-in Crystal Reports support:** You can perform ad-hoc SQL queries, display and edit the resulting (dynaset) data on TrueGrid, configure the grid (hide, resize or rearrange field columns) at runtime or design time, and generate reports on the fly.
- 28. Complete Programmer Control:** Programmer can override all default and inherited data and attributes and dictate what data, color, and font styles the grid should display in a cell using the UnboundFetch and FetchAttribute events.
- 29. Royalty Free:** TRUEGRID.VBX can be distributed free with your runtime applications.
- 30. On-Line Help:** Context sensitive help with detailed overview, property and event references, and plenty of examples.
- 31. Samples:** TrueGrid Pro comes with plenty of sample applications to illustrate product features. They also provide useful tips and tricks.

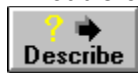
The Layout Editor Dialog

Click on a graphic element to display information about that element.



Country	Last	Birth	Birth	Death	
Row 1, Col 3	Row 1, Col 2	Row	Row 1, Col	Row 1, Col	↑
Row 2, Col 3	Row 2, Col 2	Row	Row 2, Col	Row 2, Col	
Row 3, Col 3	Row 3, Col 2	Row	Row 3, Col	Row 3, Col	
Row 4, Col 3	Row 4, Col 2	Row	Row 4, Col	Row 4, Col	
Row 5, Col 3	Row 5, Col 2	Row	Row 5, Col	Row 5, Col	
Row 6, Col 3	Row 6, Col 2	Row	Row 6, Col	Row 6, Col	
Row 7, Col 3	Row 7, Col 2	Row	Row 7, Col	Row 7, Col	↓

Double-clicking on a column in the Layout Editor will bring up the Properties pop-up.



For instructions on how to use the Layout Editor click "Describe".

Close

The Layout Editor Dialog

The Layout Editor is a tool you can use at design-time to configure many of the TrueGrid properties and gives you a WYSIWYG view of how TrueGrid will appear at runtime. You run the Layout Editor by clicking on the Layout property in the Visual Basic properties window, or by clicking on a TrueGrid control with the right mouse button.

The Layout Editor provides complete design-time control over the appearance of TrueGrid and allows you to control the following aspects of the grid:

1. **Column Configuration.** You can change the heading, fieldname, format mask, expression, column text fonts and heading fonts, column and heading justification, editability, and visibility of columns.
2. **Locked Columns.** You can lock columns at the left margin (as shown above).
3. **Splits.** You can control the size and position of splits, as well as customize the attributes of columns within each split. The ability to control splits is available only if you start the Layout Editor while SplitLocked is *False*. Normally, it defaults to *True*, so the split capabilities are disabled.
4. **Value Lists.** Using the [Field Values Dialog](#) you can design several types of values lists, including the ability to translate values. You can also have the data displayed in several formats such as in a dropdown combo, radio buttons, or allowing the user to cycle through available choices.

Moving and Resizing Columns

If columns are currently displayed in the layout editor, you can simply move them or resize them to adjust their appearance.

To move a column, simply grab the column by its heading using the mouse, then drag the column to the left or right.

To resize a column, place the mouse over the vertical separator line between columns and adjust the column size by dragging the line to the left or right. Columns are always adjusted so that they contain an integral character width, so exact placement may not be possible.

If you resize a column to zero-width, the column is made invisible. You can still access the column's properties, but you must use the [Column Properties Dialog](#) to do so.

Changing Basic Column Properties

If you double-click over a column, or if you click on a column with the right mouse button, a pop-up appears which allows you change basic column properties as well as insert and delete columns:

The popup will look like this:

<input checked="" type="checkbox"/> L eft Justify <input type="checkbox"/> C enter <input type="checkbox"/> R ight Justify
<input checked="" type="checkbox"/> E ditable
<input type="checkbox"/> H eading Text... <input type="checkbox"/> F ield Name... <input type="checkbox"/> E dit M ask...
<input type="checkbox"/> I nsert Column <input type="checkbox"/> D elete Column
<input type="checkbox"/> P roperties... <input type="checkbox"/> V alue List...

Each of the options is described below:

Left Justify, Right Justify, and Center allow you to select the justification for the column. Normally, this affects the justification for both the column text and the heading. However, if the heading justification was changed in the [Column Properties Dialog](#), then this option affects only the text justification of the column.

Editable allows you to change the editability of the column. It will be checked if the column is currently editable, and unchecked if it is not.

Selecting **Heading Text...** will allow you to change the heading text of the column. The cursor will be moved to the heading area of the column and you can change the text displayed there.

Field Name... allows you to change the field name associated with bound columns. The field name is entered in the heading area, although it will not be displayed there when you have finished changing it.

Selecting **Edit Mask...** will allow you to enter a Visual Basic Format String for the column. Like the field name, you enter the edit mask in the heading region, although it will not be displayed when you are done entering it.

Insert Column will insert a new column to the left of the current column. The newly inserted column will be a bound column, but no field name will be associated with the column initially.

Delete Column will physically delete the column from the layout. If you inadvertently delete a column, you need to exit the layout editor using the Discard option of the system menu. This will discard all changes you made during the layout editor session. When you reenter the layout editor, the column layout will be the same as it was on the original form.

The **Properties...** option will display the [Column Properties Dialog](#) and position you to the current column. This dialog is a more complete way to edit most of the column characteristics.

Clicking the **Value List...** option will bring up the [Field Values Dialog](#) and allow you to create a value list for the current column. This dialog allows you to create a value list interactively at design time.

Working With Splits in the Layout Editor

By default, TrueGrid's properties are set so that splits cannot be created directly in the layout editor. For many people, this behavior is ideal, since the "Locked at Left" option of the Column Properties Dialog will take care of creating a split automatically for locked columns.

However, advanced users may wish to use the split capabilities for more sophisticated operations. In such cases, all of the capabilities of splits are available at design time.

To enable splits in the layout editor, you must be sure that the SplitLocked property is set to *False*, which unlocks the displayed split. Now, upon entering the layout editor, a small *split create box* will appear at the leftmost extent of the horizontal scrollbar. You can drag the box interactively to create a new split.

Once splits exist in the layout editor, you can choose to have changes you make to columns be applied only to the column within the currently selected split. To do so, you need to make sure `SplitPropsGlobal` is set to *False*, which allows you to have access to split-specific behavior. Although most properties can be applied on a per-split basis, some properties (`FieldName`, `EditMask`, `Bound/Unbound/Calculated column state`) are global properties of a column. The Column Properties Dialog will automatically highlight column-global properties in blue so you know which settings apply to the column globally, and which apply on a per-split basis.

Layout Editor Caption

The caption indicates the name of the control (specified by the Name property) as well as the legend "Layout Editor" so you know which control you are editing and that you are presently in the layout editor. You can reposition the layout editor on your screen by holding the mouse down over the caption bar.

Inactive Headings

Only one split (section of the layout) is active at one time. The inactive headings are shown using the `InactiveForeColor` and `InactiveBackColor` to differentiate them from the active split. You can customize these colors on a per-split basis by leaving the layout editor, assuring that `SplitPropsGlobal` is *False*, then setting the headings while `SplitIndex` points to the proper split number.

Active Split Headings

Only one split (section of the layout) is active at one time. The active split is shown using the `HeadBackColor` and `HeadForeColor` settings for the grid. You can customize these colors on a per-split basis by leaving the layout editor, assuring that `SplitPropsGlobal` is *False*, then setting the headings while `SplitIndex` points to the proper split number.

Locked Columns

Locked columns are always at the left, and are really just a special type of split. Locked columns are easily created by using the Locked at Left checkbox in the Column Properties dialog and TrueGrid automatically sets up the split properties for you. If you master the proper split properties, you can create locked columns (or variations) yourself.

Column Within a Split

The columns within a split can be modified by double-clicking on the column (or by using the right mouse button). This split is inactive (note the heading color). You can move the columns (by holding down the mouse on their heading), or adjust their size by moving the line between columns. When you attempt to edit a column within a split, the Layout Editor automatically makes the split the active split. You can simply click on a split to make it the active split if you want. If `SplitPropsGlobal` is *False*, then each split has its own column settings; otherwise, when you change column settings, they are changed for all splits.

Column Within a Split

The columns within a split can be modified by double-clicking on the column (or by using the right mouse button). This split is the active split (note the heading color). You can move the columns (by holding down the mouse on their heading), or adjust their size by moving the line between columns. When you attempt to edit a column within a split, the Layout Editor automatically makes the split the active split. You can simply click on a split to make it the active split if you want. If `SplitPropsGlobal` is *False*, then each split has its own column settings; otherwise, when you change column settings, they are changed for all splits.

Horizontal Scroll Bar

The horizontal scroll bar allows you to scroll columns from left to right to examine them, or change their size. Each split has its own horizontal scrollbar and the position can be different for each split. The position of the scrollbars in the layout editor is not saved. When you run your application, all of the splits will be positioned to their leftmost column.

Vertical Scrollbar

Normally, there is only one vertical scrollbar (see `SplitGroup` for a discussion of multiple vertical scrolling). If you have sample data displayed in the cells (shown here), then the vertical scrollbar allows you to scroll through the sample data; otherwise, the scrollbar does nothing. The scrollbar won't be shown if you have disabled it using the `VertScrollbar` property.

Scrollbar Placeholder

Locked columns have no horizontal scrollbar, and instead display a grey area where the scrollbar would be. If the grid itself has no horizontal scrollbars at all (due to the setting of HorzScrollbar for example), then this grey area will be omitted as well.

"New Split" Box

The small black box at the leftmost side of the leftmost split (locked columns don't have them) can be used to create new splits interactively. All you need to do is drag the new split box to the right and a new split will be created. The new split inherits all of the column settings from the split immediately to its right. Unless you have `SplitLocked` set to *False*, you won't be able to create splits in the Layout Editor. Once splits are created, you can lock them using the `SplitLocked` property so the user can't change them.

"Resize Split" Box

Between splits, the small black box allows you to move the boundary between splits. Just drag it to the left or right to resize the split, or all the way to the left or right boundary to delete a split. Unless you have `SplitLocked` set to *False*, you won't be able to resize a split. Once you have set the split sizes, however, you can change `SplitLocked` so that the user can no longer adjust them.

System Menu

M ove	
C lose	Alt+F4
D iscard	
C lear Fields	
S how Sample Data	
L ayout Editor H elp	

The system menu contains several options for controlling the overall Layout editor, including options for displaying sample data, clearing the grid, and discarding changes before returning to Visual Basic.

"Move" Option

The Move option allows you to move the layout editor anywhere on your screen. You can also move the layout editor by grabbing the caption area.

"Close" Option

The Close option will close the layout editor, save all changes you have made to the layout, and return to Visual Basic.

"Discard" Option

Choosing the Discard option will ignore any changes you have made during this Layout Editor session and return to the Visual Basic design environment. The original layout will remain intact in the TrueGrid control on your form.

"Clear Fields" Option

Selecting Clear Fields will remove all splits and columns, making the current layout completely blank. If you then choose "Close", all layout information will be deleted and the grid will return to its initial state with no design-time layout at all. If you inadvertently choose "Clear Fields", then just choose Discard and return to Visual Basic; the original control layout will remain intact.

"Show Sample Data" Option

This option will toggle the layout editor display of sample data. The sample data is useful for showing how text formatting within columns will look. It also displays the row and column numbers in the columns, which is useful for knowing exactly what the column indexes are for use in your code.

"Layout Editor Help" Option

This option will bring up the Layout Editor Help screen.

Column Properties Dialog

Click on a graphic element to display information about that element.

Column Properties

Column: 1

Bound Unbound Calculated

Heading:

Width: Limit:

Format\$:

Field Name:

Expression:

Heading

Bold

Italic

Underline

Locked at left

Visible

Text

Bold

Italic

Underline

Editable

Preserve

For instructions on how to use the Column Properties Dialog click "Describe".

Close

Column Properties Dialog

Introduction to the Column Properties Dialog

The column properties dialog was added in TrueGrid 2.0 to make it easier to change many of the column properties. You invoke the column properties dialog by bringing up the [Properties Pop-up](#) in the [Layout Editor](#). The column you clicked on to bring up the properties popup will be the initial column displayed in this dialog.

Any changes you make to the current column can be discarded by clicking Cancel (which returns you to the Layout Editor). Clicking OK will save the changes you made. You can move from column to column using the scrollbar at the left, and insert new columns with New as well as delete existing columns with Delete. The Update button will update the changes to the current column without leaving the properties dialog.

Once changes are made to a column, and updated, the changes are immediately reflected in the layout grid. If you make changes to properties which affect the appearance of text in columns, you may want to be sure you select Show Sample Data from the system menu of the Layout Editor. TrueGrid will fill the grid with sample data so you can see the effect of your changes.

Properties Affected by the Dialog

The column properties dialog controls the following properties:

ColumnExpression	Corresponds to the expression typed in the Expression area of the dialog box.
ColumnField	Choosing a fieldname in the Field Name combo, or typing one in, changes this property.
ColumnFontStyle	Clicking on the text Bold , Underline , or Italic checkboxes changes this property from -1 (meaning the text font is inherited from the grid) to the appropriate font style.
ColumnHeadFontStyle	Clicking on the heading Bold , Underline , or Italic checkboxes changes this property from -1 (meaning the heading font is inherited from the grid) to the appropriate font style.
ColumnName	Corresponds to the value typed in the Heading field.
ColumnSize	Changed by setting the value of the Limit entry in the dialog.
ColumnStyle	The ColumnStyle property contains bits which indicate the disposition of the column justification and editability. These bits correspond to the Heading Justification and Text Justification combos as well as the Editable checkbox.
ColumnType	Changed by clicking on the Bound , Unbound , or Calculated column radio buttons.
ColumnVisible	The Visible checkbox will set this property.
ColumnWidth	Changed by setting the value of the Width entry in the dialog.
EditMask	Changing the Format\$ entry sets this property.

There is no property corresponding to the **Locked at Left** checkbox (see below).

Locking Columns at the Left Margin

The Locked at Left checkbox is a convenience feature designed to simplify the use of splits. Locking columns is a common use for splits, and since several properties must be set to accomplish this, the Locked at Left process automates a common task.

When you click Locked at Left, TrueGrid will automatically create a split and set the proper properties so that the selected column appears in the leftmost part of the grid. Once locked, you can treat the column just like any other, changing its width and other characteristics without concern for its locked status.

To unlock the column, just uncheck the box.

How "Locked at Left" Works

Once you master the split properties (see the advanced user's overview for Realizing the Full Potential of Splits) you can create more sophisticated effects than the simple locked column. It may be useful, in such cases, to know exactly what "Locked at Left" does.

When you lock a column, TrueGrid does the following:

1. If there is no split at the left, TrueGrid creates one.
2. TrueGrid assures that all of the columns which are locked have `ColumnVisible` set to `True` in the leftmost split, and have `ColumnVisible` set to `False` in all of the remaining splits. Note that all columns are always present in all splits, but it is the visibility state of the column that makes it appear where you want it.
3. TrueGrid makes sure that the leftmost split has its `SplitLocked` property set to `True` so that it is a fixed split and the user cannot remove or resize it.
4. TrueGrid sets the `SplitSizeMode` of the leftmost split to "2 - Number of columns" so that the split will always conform to the size of an integral number of columns.
5. TrueGrid sets the `SplitSize` of the leftmost split to equal the exact number of locked columns. This causes TrueGrid to display only the locked columns and to keep the split at the proper size.

So long as the above properties remain consistent, TrueGrid will continue to support the "Locked at Left" behavior in the Layout editor. If you modify the `SplitSizeMode`, `SplitSize`, or `SplitLocked` setting for the left split, or if the locked columns are not invisible in all other splits, TrueGrid assumes that you know what you are doing and the "Locked at Left" checkbox will be grayed-out.

Although there is no corresponding property at runtime, the utility functions `TgLockColumnLeft`, `TgLockColumnRight`, and `TgUnlockColumn` can be used to manipulate locked columns at runtime.

More About Splits

By default, you don't need to know much about splits, but once you learn how to use them, TrueGrid enables you to control many split characteristics in the column properties dialog.

The column properties dialog is affected by splits in two ways:

1. When you enter the column properties dialog, not only are you positioned at a particular column, but if you have multiple splits, you are positioned within a particular split. You can tell which split you are in by looking at the TrueGrid headings in the layout. The current split will use the active heading color while all other splits will use the inactive heading color.
2. Normally, changing column properties will change the properties in all splits. However, if you have `SplitPropsGlobal` set to `False`, then TrueGrid allows you to make changes to columns in only the current split. To make this clearer, the column properties dialog will change slightly if `SplitPropsGlobal` is `False` to indicate which settings are global, and which are applied only to the current split.

Column Number

The column number shows the column currently being edited. When you click on the layout editor to bring up the properties pop-up, the column you click over will be the column initially displayed. Use the scrollbar at the left to move from column to column.

Scrollbar for Changing Columns

Use the scrollbar at the left to change from column to column. When moving to another column, information for the current column will be automatically updated to the grid displayed in the layout editor. If you want to discard information you've changed, then click Cancel instead.

Bound, Unbound, or Calculated Column Radio Buttons

Bound columns require a field name to be specified, which will automatically be fetched and updated by TrueGrid. For unbound columns, you need to write code in the UnboundFetch event to fill the column with data when it is needed. When calculated columns are selected, an expression can be entered at the bottom of the dialog. TrueGrid will automatically calculate the expression and fill the column with data for you. These radio buttons correspond to setting the ColumnType property in your code at runtime.

Heading Text

The heading field will be displayed at the top of the column. The heading can be as long as you like and will wrap within the heading area if HeadingHeight has been set large enough to accommodate more than one row. The heading text can be changed at runtime using the ColumnName property.

Column Display Width

The Width setting specifies the size of the column as it is displayed. This value is specified in characters. Changing this value will change the actual displayed width of the column in the layout editor. You can also change the width by moving the vertical lines in the [Layout Editor](#). At runtime, you can inspect and change this value using the ColumnWidth property.

Column Data Entry Limit

This setting specifies the maximum number of characters which can be entered into the field. It is independent of the column's display width. The default value (if non-zero) is derived from the database definition. If you set this value to zero, then TrueGrid imposes no data entry limit. At runtime, you can inspect and change this value using the ColumnSize property.

Format\$ String

This combo box specifies the Visual Basic Format String to be used for *displaying* data in the column. This does not affect the storage format of the data, but only the displayed value. The drop-down contains a list of frequently used formats. However, you can type directly into the text area to enter your own format string, or to clear the format string for the column. At runtime, this setting can be changed and inspected using the EditMask property.

Column Field Name

This item is available only for bound columns. It contains the database field name to which the column is bound. If TrueGrid is linked to a data control, then the drop-down portion will contain a list of valid fieldnames for you to choose from. However, you can type directly into the text area of the combo if you know the fieldname, or if the fieldname will be valid at runtime even though it is not available at design time. At runtime, this setting can be changed and inspected using the ColumnField property.

Calculated Expression

This item is available only for calculated columns. When enabled, you must enter a valid Visual Basic Expression consisting of Visual Basic Operators and functions supported by TrueGrid. When you update the column information, TrueGrid checks the expression for validity. At runtime, you can inspect and change this value by using the ColumnExpression property.

Heading Justification Combo

This combo box allows you to select Left, Right, or Centered justification styles for the current column heading. At runtime, this setting can be changed and inspected using the `ColumnStyle` property.

Column Text Justification Combo

This combo box allows you to select Left, Right, or Centered justification styles for the current column text. In order to see the effect of this setting, you need to turn the sample data display in the Layout Editor system menu. At runtime, this setting can be changed and inspected using the ColumnStyle property.

Heading Font Bold

This checkbox applies bold font characteristics for the heading of the current column. If the checkbox is grayed-out, then all font characteristics are currently inherited from the default grid font characteristics. Checking either the bold, italic, or underline boxes will force the column to override default settings and all three boxes will no longer be grayed-out. At runtime, this setting can be changed and inspected using the `ColumnHeadFontStyle` property.

Heading Font Italic

This checkbox applies italic font characteristics for the heading of the current column. If the checkbox is grayed-out, then all font characteristics are currently inherited from the default grid font characteristics. Checking either the bold, italic, or underline boxes will force the column to override default settings and all three boxes will no longer be grayed-out. At runtime, this setting can be changed and inspected using the `ColumnHeadFontStyle` property.

Heading Font Underline

This checkbox applies underlined font characteristics for the heading of the current column. If the checkbox is grayed-out, then all font characteristics are currently inherited from the default grid font characteristics. Checking either the bold, italic, or underline boxes will force the column to override default settings and all three boxes will no longer be grayed-out. At runtime, this setting can be changed and inspected using the `ColumnHeadFontStyle` property.

Column Text Font Bold

This checkbox applies bold font characteristics for the text of the current column. The change will not be apparent in the grid unless the Show Sample Data option is selected in the Layout Editor system menu. If the checkbox is grayed-out, then all font characteristics are currently inherited from the default grid font characteristics. Checking either the bold, italic, or underline boxes will force the column to override default settings and all three boxes will no longer be grayed-out. At runtime, this setting can be changed and inspected using the ColumnFontStyle property.

Column Text Font Italic

This checkbox applies italic font characteristics for the text of the current column. The change will not be apparent in the grid unless the Show Sample Data option is selected in the Layout Editor system menu. If the checkbox is grayed-out, then all font characteristics are currently inherited from the default grid font characteristics. Checking either the bold, italic, or underline boxes will force the column to override default settings and all three boxes will no longer be grayed-out. At runtime, this setting can be changed and inspected using the ColumnFontStyle property.

Column Text Font Underline

This checkbox applies underlined font characteristics for the text of the current column. The change will not be apparent in the grid unless the Show Sample Data option is selected in the Layout Editor system menu. If the checkbox is grayed-out, then all font characteristics are currently inherited from the default grid font characteristics. Checking either the bold, italic, or underline boxes will force the column to override default settings and all three boxes will no longer be grayed-out. At runtime, this setting can be changed and inspected using the ColumnFontStyle property.

Locked At Left Setting

Checking this box causes the current column to be locked at the leftmost portion of the grid. In actuality, TrueGrid creates a split at the left of the grid and automatically sets the proper properties to lock this column. Unchecking the box will unlock the column. If this box is greyed, then you have modified the layout of splits so that TrueGrid can no longer recognize the left split as the one containing locked columns. There is no property equivalent for this checkbox.

Column Visible Checkbox

If this box is checked, then the current column is visible in the grid. If unchecked then the column will not be visible. Upon exiting the layout editor, any invisible columns are physically removed from the layout rather than being stored. (unless Preserve is also checked). At runtime, column visibility can be controlled using the ColumnVisible property.

Preserve Checkbox

Checking this box will preserve a column as part of the design-time layout, even if it is invisible in all splits of the grid. Normally, upon exit from the Layout Editor, any columns which are completely invisible are removed from the layout to conserve runtime memory. Sometimes, it is useful to preserve hidden columns for some runtime purpose (such as unbound or calculated columns which depend upon their values). If a column is visible, this checkbox is grayed-out.

Column Editable Checkbox

If this box is checked, then the current column will be editable by the user. If unchecked, no editing can take place in the column. At runtime, this setting may be changed by using the `ColumnStyle` property. Even if a column is marked as editable, editing will not be allowed if the `Editable` property is set to *False* for the grid, or if the column's split is marked as uneditable with the `SplitEditable` property.

Create New Column Button

Clicking on this button will insert a new column immediately to the left of the the current column. Any changes to the current column will be saved before the new column is created.

Delete Current Column Button

Clicking on this button will delete the current column from the layout. If the column occurs in more than one split, it will be removed from all splits simultaneously. Any changes to the current column will be lost. If you want a column to be visible in one split and not in another, don't use the Delete button, but rather make the column invisible in the splits where you do not want it to appear. Split-specific column changes are only possible if `SplitPropsGlobal` is set to *False*.

Update Column Properties Button

Clicking on this button will update the properties for the current column so that they are reflected in the layout editor grid. Moving to a new column, adding a new column, or clicking OK will also update the current column properties.

Cancel Current Column Edits

Clicking on this button will discard any changes made to the current column only and return to the layout editor grid. Any changes made to other columns will still be permanent unless the Discard option is used in the [Layout Editor](#) system menu.

Help Button

Clicking on this button will bring up the on-line help page for the column properties dialog. Help can also be obtained by pressing F1 in the column properties dialog or in the layout editor.

OK Button

Clicking on this button will save the current column definition and return to the layout editor grid.

Value list.. button

Clicking this button will bring up the Field Values Dialog and allow you to create a value list for the current column. This dialog allows you to create a value list interactively at design time.

Properties Pop-up (Layout Editor)

<input checked="" type="checkbox"/> L eft Justify <input type="checkbox"/> C enter <input type="checkbox"/> R ight Justify
<input checked="" type="checkbox"/> E ditable
<input type="checkbox"/> H eading Text... <input type="checkbox"/> F ield Name... <input type="checkbox"/> Edit M ask...
<input type="checkbox"/> I nsert Column <input type="checkbox"/> D elete Column
<input type="checkbox"/> P roperties... <input type="checkbox"/> V alue List...

The properties pop-up can be accessed by double-clicking on a column in the layout editor, or by clicking on a column with the right mouse button. Each of the entries in this dialog corresponds to entries in the Column Properties Dialog but they are included here as well for convenience.

Left Justify (pop-up)

Allows you to select left justification for the column. Normally, this affects the justification for both the column text and the heading. However, if the heading justification was changed in the Column Properties Dialog, then this option affects only the text justification of the column.

Center (pop-up)

Allows you to select centered text for the column. Normally, this affects the justification for both the column text and the heading. However, if the heading justification was changed in the Column Properties Dialog, then this option affects only the text justification of the column.

Right Justify (pop-up)

Allows you to select right justified text for the column. Normally, this affects the justification for both the column text and the heading. However, if the heading justification was changed in the Column Properties Dialog, then this option affects only the text justification of the column.

Editable (pop-up)

The editable option allows you to change the editability of the column. It will be checked if the column is currently editable, and unchecked if it is not.

Heading Text... (pop-up)

This option will allow you to change the heading text of the column. The cursor will be moved to the heading area of the column and you can change the text displayed there.

Field Name... (pop-up)

Selecting this option allows you to change the field name associated with bound columns. The field name is entered in the heading area, although it will not be displayed there when you have finished changing it.

Edit Mask... (pop-up)

Selecting **Edit Mask...** will allow you to enter a Visual Basic Format String for the column. Like the field name, you enter the edit mask in the heading region, although it will not be displayed when you are done entering it.

Insert Column (pop-up)

Insert Column will insert a new column to the left of the current column. The newly inserted column will be a bound column, but no field name will be associated with the column initially.

Delete Column (pop-up)

This option will physically delete the column from the layout. If you inadvertently delete a column, you need to exit the layout editor using the Discard option of the system menu. This will discard all changes you made during the layout editor session. When you reenter the layout editor, the column layout will be the same as it was on the original form.

Properties... (pop-up)

The **Properties...** option will display the Column Properties Dialog and position you to the current column. This dialog is a more complete way to edit most of the column characteristics.

Value List... (pop-up)

Clicking the **Value List...** option will bring up the Field Values Dialog and allow you to create a value list for the current column. This dialog allows you to create a value list interactively at design time.

Field Values Dialog

Click on a graphic element to display information about that element.

Data	User Value
1	Author
2	Publisher
3	Reviewer

Display values:

as Text

as Radio Buttons

as Combo Box

Sorted

Translate Values

Auto Validate

Cycle on Click

Default Value

OK Cancel Picture...



For instructions on how to use the Field Values dialog click "Describe".

Close

Field Values Dialog

Version 2.1 of TrueGrid adds a powerful new feature called "value list". Using this feature you can dramatically enhance the display of the grid. At design time you can set up these features using the Field Value Dialog. To access the Field Values dialog click on the Value List option from the Properties pop-up or click the Value List... button in the Column Properties Dialog. The column you clicked on to bring up the properties popup will be the initial column displayed in this dialog.

Any changes you make to the current column's value list can be discarded by clicking Cancel (which returns you to the previous dialog). Clicking OK will save the changes you made.

Once you are in this dialog you can specify the valid values for a column in the data column. A common simple use is to check the "cycle on click" check box in which case when the user clicks on that column it will display each value in the data column in succession. You could also check the "Auto Validate" check box and the user would then be able to enter only values specified in the data column.

Once the data is entered you then need to choose how you want to display the data. You can click on one of the three radio buttons to choose how you want the data displayed. If you select the "Text" button the data will be displayed normally. If you check the "Radio buttons" option then the grid will take the values in the data column and display them as radio buttons. Selecting the "Combo" option will result in a dropdown combo box being automatically displayed with the values taken from the data column. If you check the "Sorted" check box in conjunction with the "Combo" option then the values will be displayed in sorted order.

If you want a certain value to be displayed even if it is not in the value list then select that value and click the "Default" check box. This value will be displayed for any value not listed in the data column.

Translated values and pictures

Using value list to automate the data entry is great but one of the most important features that TrueGrid enables you to do is substitute database data with your own! If you click on the "Translate Values" check box you will notice that another column appears in the grid with the heading "User value". Using this column it is possible to substitute cryptic values such as 0 and -1 with True/False, Yes/No, or whatever you heart desires. Neat huh? It gets better. Let's say that you want to show a simley face and a frowning face. The toughest part is getting the bitmaps (we stole them from WinMine).

Once you have the bitmaps simply select the row you want the picture to appear in and click the "Picture..." button. A file open dialog appears and you can select your bitmap. When you return the bitmap will be displayed in the right hand side of the grid. If you choose to use pictures then it is highly recommended that you use the Text option for display. You can still use the "Cycle on click" option, in fact this is recommended when using things like check boxes. However, using AutoValidate doesn't make much sense since the user can't "enter" the picture.

Properties Affected by the Dialog

The field values dialog controls the following properties:

ColumnButton	Automatically set to <i>True</i> if the combo option is selected.
VlistCount	Automatically set to the number of values in the data column.
VlistData	An indexed property that corresponds to the values in the data column.
VlistDefault	Correspondes to the Default check box.
VlistPicture	An indexed property that corresponds to any pictures used in the user value column.
VlistStyle	Set using a combination of the display values frame, and the cycle on click, auto validate, and translate values check boxes.
VlistText	An indexed property that corresponds to the text values used in the user value column.

Layout Editor Caption

The caption indicates the name of the column (specified by the ColumnName property) as well as the legend "Field Values..." so you know which column you are editing and that you are presently in the field values dialog. You can reposition the field values dialog on your screen by holding the mouse down over the caption bar.

Data Value Column

This column holds the values that represent the values in the database. In this column you should specify all the values you want the user to be able to enter (if you click Auto Validate), or cycle through (if you click Cycle on Click). If the Translate values check box is turned off these are the values that will be displayed in the grid at run-time.

User Value Column

This column holds the value the programmer wants the user to see. These values can either be text or pictures. This column is only present when the "Translate Values" check box is turned on. In this column you should specify all the values you want the user to be able to enter (if you click Auto Validate), or cycle through (if you click Cycle on Click).

Data Column Entries

These are the values that TrueGrid will compare against the database (or array) to either substitute with the translated value, auto validate, or cycle through, depending on which options are turned on. The number of entries can be checked at run-time using the VlistCount.

User Value Entries

These are the values that TrueGrid will display in the grid in place of it's corresponding value in the data column. These values can be either text or pictures. You can check these values at run time using the VlistText or VlistPicture depending on contents of the column.

OK Button

Clicking on this button will save the current value list definitions and return to the layout editor or the column properties dialog (depending on which one you entered the dialog from).

Cancel Button

Clicking on this button will discard any changes made to the current value list and return to the layout editor or column properties dialog (depending on which one you entered the dialog from).

Picture Button

Clicking this button allows you to have BMP files displayed in the User value column. When you click this button a File Open dialog will pop up and allow you to select a BMP file. The picture will then appear in the current row in the user column.

Display as Text

This is the default display method. If you have this option chosen then the grid will display the Data value or the User value, depending on the Translate Values check box, in standard format.

Display as Radio Button

This option allows you to display a set of Radio buttons in place of the standard text layout. In each cell there will be a radio button for each option listed in the data or user value column.

Display as Combo Box

Using this option the grid will display the data in a standard text layout. However if the user clicks on a column with this setting a ColumnButton and an automatic dropdown combo will appear for the user to select from. You can control the number of items displayed in the dropdown before a scroll bar appears by setting the ColumnComboMaxItems

Sorted Check Box

This check box is only enabled when the Combo option is selected. If Sorted is checked the values that appear in the dropdown combo will be in sorted order.

Translate Values Check Box

If you want the values in the data column to be substituted with different values then check the translate check box. When checked a second column appears in the grid and allows you to enter in values to be displayed in the grid at run time in place of the values in the data column.

Auto Validate Check Box

If checked then the user will only be able to enter in values from the value list. If the user tries to enter in a value that is not in the list then the grid will beep and stay in edit mode until the user enters a valid value.

Cycle on Click Check Box


If checked the users can cycle through all the values in the value list simply by clicking on the cell. Also the user will only be able to edit the cell by explicitly typing in a value.











































Default Value Check Box

If checked the current column in the value list will change the font to an italic. This value will then be used for any value that is in the database or typed in by the user that is not in the value list.

Property and Event Reference by Area

 Overall grid color

 Colors specific to splits

  Describe	Colors specific to columns within splits
  Describe	Overall grid fonts
  Describe	Fonts specific to splits
  Describe	Fonts specific to columns within splits
  Describe	Per-cell colors and fonts
  Describe	Controlling margins within cells
  Describe	Heading size and presence
  Describe	Horizontal and vertical grid lines
  Describe	Miscellaneous grid appearance properties
  Describe	Row and column numbering
  Describe	Calculated, bound, and unbound columns
  Describe	Miscellaneous column information
  Describe	Scrollbars
  Describe	Layout configuration
  Describe	User keyboard and mouse navigation
  Describe	User modification of grid appearance
  Describe	Editing cell contents
  Describe	Controlling database behavior
  Describe	Working with multiple row selections
  Describe	Controlling splits
  Describe	Drag and drop



Determining cell dimensions and location



Finding cells based upon coordinates



Programmer control



Value list



Column Summary

Group: Overall grid color

These properties control the overall color of the grid. They are the default colors used by all splits, and all columns of the grid. These can be overridden by changing split-specific, column specific, and cell-specific colors. These color values are not split-specific, but apply globally.

Properties

BackColor	Changes the background color of the grid to the specified color value. Can be changed at design time or runtime.
ForeColor	Changes the foreground color of the grid to the specified color value. Can be changed at design time or runtime.

Group: Colors specific to splits

TrueGrid allows certain "basic" colors to be set to control default colors for the headings (whether active or inactive), marked rows, vertical and horizontal lines, and the color of a cell when it is in editing mode. These properties can be set at design time or runtime. If changed, they take effect immediately.

These properties are affected by the setting of SplitPropsGlobal. If it is *False*, then the setting will apply only to the current split; if *True* it applies to all splits.

Properties

EditBackColor	Controls the background color of a cell when it is opened for editing. By default, this property is set to the system highlight background color. Because of this, text selected with the mouse (although still selected) will not be evident. Change this property (and EditForeColor) if you want to differentiate highlighted text.
EditForeColor	Controls the foreground color of a cell when it is opened for editing.
HeadBackColor	Specifies the background color of the headings whenever the grid is active. When inactive, the InactiveBackColor is used for the headings. If there are multiple splits, this color is used only for the current split.
HeadForeColor	Specifies the foreground color of the headings whenever the grid is active. When inactive, the InactiveForeColor is used for the headings. If there are multiple splits, this color is used only for the current split.
HorzColor	Specifies the color used for horizontal lines (if present). If the HorzLines property is set to 3D mode, then this color specifies the color of the darker border of 3D bevels; the lighter color is derived from this.
InactiveBackColor	Controls the background color of the headings whenever the grid (or split) is inactive.
InactiveForeColor	Controls the foreground color of the headings whenever the grid (or split) is inactive.
SelectedBackColor	If a row is currently selected by marking the row, this color specifies the background color for that row. By default, the color is black. Selected rows are shown only if SelectMode is set accordingly. This color can be overridden by cell-specific color settings.
SelectedForeColor	If a row is currently selected by marking the row, this color specifies the foreground color for that row. By default, the color is black. Selected rows are shown only if SelectMode is set accordingly. This color can be overridden by cell-specific color settings.
VertColor	Specifies the color used for horizontal lines (if present). If the HorzLines property is set to 3D mode, then this color specifies the color of the darker border of 3D bevels; the lighter color is derived from this.

Group: Colors specific to columns within splits

Each column can have its own default background and foreground color, specified by these properties. These properties can be set only at runtime and can be overridden by cell-specific color settings if desired. These colors accept a special value, `INHERIT_COLOR` (defined in `TGCONST.TXT`) which indicates that the column color is to be inherited from the default grid color.

These properties are affected by the setting of `SplitPropsGlobal`. If it is *False*, then the setting will affect the column only within the current split; if *True* it applies to the column in all splits.

Properties

<code>ColumnBackColor</code>	Specifies the background color of the column. The default setting is <code>INHERIT_COLOR</code> , indicating that the column has no special background color.
<code>ColumnForeColor</code>	Specifies the foreground color of the column. The default setting is <code>INHERIT_COLOR</code> , indicating that the column has no special foreground color.

Group: Overall grid fonts

TrueGrid allows customization of fonts in a variety of circumstances. However, the basic font face (specified by `FontName`) and the point size of the font (`FontSize`) are global to the entire grid and cannot be changed.

These properties are not affected by `SplitPropsGlobal`, and control the base font used for all cells and headings in the grid.

Properties

<code>FontBold</code>	If <i>True</i> then the base font will be bold. This is the default.
<code>FontItalic</code>	If <i>True</i> then the base font will be italic. Defaults to <i>False</i> .
<code>FontName</code>	Specifies the font typeface name to be used for all grid cells and headings. This font cannot be overridden and must apply globally to the entire grid. However, the bold, italic, and underline properties of the font (referred to as the <i>font style</i>) can be changed selectively on a per-column or per-cell basis, as well as for the headings.
<code>FontSize</code>	This property specifies the point size for fonts used within the entire grid and cannot be overridden. The setting of this property determines the height of a line of text, and is the basis for the heading height and row height.
<code>FontStrike</code>	If <i>True</i> then the base font will use strike-out text. This setting is global and cannot be overridden. Defaults to <i>False</i> .
<code>FontUnder</code>	If <i>True</i> then the base font will be underlined. This setting is global, and can be overridden on a column or per-cell basis. Defaults to <i>False</i> .

Group: Fonts specific to splits

The headings may use a different font style than the rest of the grid. This property is affected by the setting of `SplitPropsGlobal`. If it is *False*, then the property applies to the headings in the current split only; if *True* then all headings are affected. This property is available at design-time and runtime.

Properties

HeadFontStyle	Specifies the font style to be used for headings. Defaults to 0, meaning that the font style is inherited from the global grid font settings.
---------------	---

Group: Fonts specific to columns within splits

The font style can be specified on a per-column basis and can be used to override the base font style. These properties can be set at runtime, or at design-time in the layout editor. These font styles accept a special value, `INHERIT_FONT` (defined in `TGCONST.TXT`) which indicates that the column font is to be inherited from the default grid font.

These properties are affected by the setting of `SplitPropsGlobal`. If it is *False*, then the setting will affect the column only within the current split; if *True* it applies to the column in all splits.

Properties

- `ColumnHeadFontStyle` Specifies the font style to be used for a heading for the specified column. Defaults to `INHERIT_FONT`.
- `ColumnFontStyle` Specifies the font style to be used for cell text within the specified column. Defaults to `INHERIT_FONT`.

Group: Per-cell colors and fonts

TrueGrid supports an advanced technique, tailored to database usage, which permits colors and fonts to be specified on a per-cell basis. Rather than "apply" colors and fonts to specific cells, TrueGrid has methods for defining cell colors based upon a cell's status or contents. This makes font and color selection much faster, since TrueGrid has a better knowledge of how fonts and colors are intended to be used.

A special method (using the FetchAttributes event) allows cell color and font to be specified using any desirable means. All of these properties use the special values INHERIT_FONT and INHERIT_COLOR (defined in TGCONST.TXT) to specify that no color or font override is provided for a specific case.

The use of these properties is somewhat complex, but they provide great power and flexibility.

All of these properties are affected by the setting of SplitPropsGlobal. If it is *False*, then the setting will affect the column only within the current split; if *True* it applies to the column in all splits.

Meaning of Cell Status Values

There are 16 separate cell status values, which indicate the disposition of a cell. The status is a combination of four separate conditions:

1. *Current cell.* The cell is the current cell, specified by the ColumnIndex and RowIndex properties.
2. *Part of a highlighted row marquee.* When the MarqueeStyle property indicates that the entire row is to be highlighted, cells in such rows have this additional condition set.
3. *Updated data.* The data in the cell has been updated by user modification, or setting the Text or ColumnText properties.
4. *Marked row.* The cell is part of a row marked by the user (or programmer) and displayed as such according to the setting of the SelectMode property.

The cell status is a combination of these four conditions and there are a total of 16 different cell dispositions, shown in the following table:

Status Value	Condition of Cell			Current Cell	Meaning of Status Value
	Row Marked	Data Changed	Row Marquee		
0					Normal cell, not current, marked, changed, or part of a highlighted row.
1				X	The current cell, not marked, changed or part of a highlighted row.
2			X		Part of a highlighted row, not current, marked or changed.
3			X	X	Current cell in highlighted row, not marked or changed.
4		X			Changed cell, not current, marked, or part of a highlighted row
5		X		X	Current cell when changed, not marked or part of a highlighted row.
6		X	X		Cell changed within a highlighted row, not current or marked.
7		X	X	X	Current cell changed within a highlighted row, not marked.
8	X				Row within a marked cell, not current, changed, or part of a highlighted row.
9	X			X	Current cell within a marked row, not changed or part of a highlighted row.
10	X		X		Cell in a marked row which is also the highlighted row, not current cell and not changed.
11	X		X	X	Current cell in a marked row which is also the highlighted row, but not changed.
12	X	X			Cell within a marked row in which data has been changed, not current and not part of a highlighted row.

13	X	X		X	Current cell within a marked row in which data has been changed; not part of a highlighted row.
14	X	X	X		Cell within a marked row (which is also highlighted) in which data has been changed, but not the current cell.
15	X	X	X	X	Current cell within a marked row which is also highlighted. Data in the cell has been changed.

Each status value is distinct, and you can set a different set of colors and fonts for each of the 16 status conditions. That is, setting attributes for status 1 affects the current cell, but only when it is not marked, is not changed, and is not part of a highlighted row marquee. To affect the current cell under all circumstances, you would need to set the attributes for status values 1, 3, 5, 7, 9, 11, 13, and 15. Usually, this is not necessary since no application will have cells of all possible status values. For example, if the marquee is set to a dotted cell, then none of the status values which indicate a row marquee condition will ever be encountered.

Parameter Properties

These properties do not actually change the grid appearance, but are used as parameters for other cell-specific color and font properties. Usually, these properties must be set before using the cell-specific properties such as `SetStatusAttr`, `ColumnSetStatusAttr`, `AddRegexAttr`, and `ColumnAddRegexAttr`.

TrueGrid never modifies these properties, but initializes them all to a default value of `INHERIT_COLOR` or `INHERIT_FONT`. The properties are available only at runtime, and are global to the entire grid (even though the attribute setting properties are specific to the current split).

<code>ParamBackColor</code>	Specifies the background color to be applied to the cells specified by subsequent setting of one of the per-cell attribute properties. Defaults to <code>INHERIT_COLOR</code> .
<code>ParamFontStyle</code>	Specifies the font style to be applied to the cells specified by subsequent setting of one of the per-cell attribute properties. Defaults to <code>INHERIT_FONT</code> .
<code>ParamForeColor</code>	Specifies the foreground color to be applied to cells specified by subsequent setting of one of the per-cell attribute properties. Defaults to <code>INHERIT_COLOR</code> .
<code>ParamStatus</code>	Specifies the cell status of cells affected by regular expressions added with <code>AddRegexAttr</code> and <code>ColumnAddRegexAttr</code> . There are 16 different cell status values. This property defaults to the special value, -1, meaning that the regular expression applies to cells of any status.

Setting Per-Cell Attributes Based Upon Status

The following two properties allow you to apply the values of `ParamForeColor`, `ParamBackColor`, and `ParamFontStyle` to cells which have a given status. These properties are available only at runtime and are write-only.

<code>SetStatusAttr</code>	Applies the attributes specified by <code>ParamForeColor</code> , <code>ParamBackColor</code> , and <code>ParamFontStyle</code> to all cells with a specified status. The attributes are applied as cells are displayed, so once this property is set, the grid will respect the attributes as the grid is scrolled, or data is changed and refreshed.
<code>ColumnSetStatusAttr</code>	Like <code>SetStatusAttr</code> , but applies the attributes only to cells within a given column which have the specified status. Overrides any settings provided with <code>SetStatusAttr</code> .

Setting Per-Cell Attributes Based Upon Cell Contents

TrueGrid allows you to specify attributes which are applied to a cell based upon whether the cell matches a particular *regular expression*. A regular expression specifies a pattern which is tested against the cell contents whenever it is displayed or changed. If the pattern matches, then the specified attributes are applied. TrueGrid allows you to add an unlimited number of patterns with corresponding attributes.

<code>AddRegexAttr</code>	Adds a regular expression to the list of patterns for the current split (or all splits)
---------------------------	---

if *SplitPropsGlobal* is *True*). *ParamForeColor*, *ParamBackColor*, and *ParamFontStyle* must be set before assigning to this property. The parameters specify the attributes which are applied when a pattern match occurs. If *ParamStatus* is set to -1, then the pattern applies to all cells, regardless of status. If *ParamStatus* contains a value from 0 to 16, then only cells of that status which match the pattern are affected.

- ColumnAddRegexAttr** Like *AddRegexAttr*, but applies only to the specified column. You may set patterns with both *AddRegexAttr* and *ColumnAddRegexAttr*. Those set with *ColumnAddRegexAttr* take priority over those which are set with *AddRegexAttr* whenever a match occurs on both patterns.
- ColumnHasRegexAttr** Returns *True* if the specified column has pattern attributes associated with it. Setting it to *False* clears any pattern attributes for the column.
- HasRegexAttr** Returns *True* if the current split has pattern attributes associated with it. Setting this property to *False* clears any pattern attributes for the current split (or all splits if *SplitPropsGlobal* is *True*).

Setting Cell Colors and Fonts Using User-Defined Criteria

For cases where status-dependent or pattern-dependent attribute settings are inappropriate, TrueGrid can query your program each time it needs to know cell attributes. This is done via the *FetchAttributes* event. The event will be triggered only for those columns which have the *ColumnCellAttrs* property set to *True*. Using this capability, you can have cell colors and fonts which are dependent upon the contents of a cell (if it is a certain range, for example) or even dependent upon the contents of other cells and databases.

- ColumnCellAttrs** If set to *True* for a column, then TrueGrid will trigger the *FetchAttributes* event each time it needs to know what the cell attributes will be. TrueGrid optimizes this behavior so it calls *FetchAttributes* only when cells are scrolled into view, or when their contents or status changes.
- FetchAttributes Event** This event is fired whenever TrueGrid needs to know the color and font to use for a cell within a column which has *ColumnCellAttrs* set to *True*.

Group: Controlling margins within cells

TrueGrid allows you to change the left and right margin of each cell based upon any criteria you specify. The cell margins are dynamic. Each cell can have its own margins, and the margins can be changed as data changes. This makes it possible to implement outlining features by indenting cells according to an outline level (usually stored in the database as well) or some other criteria.

Cell margins are specific to splits and can be enabled or disabled on a per-split basis. `SplitPropsGlobal` is respected. If `SplitPropsGlobal` is *True*, then turning on cell margins for a column affects all splits; otherwise, only the current split is affected.

Properties

`ColumnFetchMargins` If *True*, then the specified column will have cell margins, which causes the `FetchMargins` event to be triggered each time the grid needs to know cell margins within the column. Defaults to *False* for all columns.

Events

`FetchMargins` If `ColumnFetchMargins` is *True* for a column, this event will be triggered each time the grid needs to know the cell margins (usually at the same time it fetches the cell data). The cell margins are specified by the `InsetLeft` and `InsetRight` arguments, which are changed to reflect the new margins. Margins are specified in 10ths of a character, so that the margins are proportional to the font being used.

Group: Heading size and presence

TrueGrid allows you to have headings for columns, or to hide them. These properties can be set at design time or runtime. If changed, they take effect immediately.

Properties

Headings	If <i>True</i> then column headings are displayed; if <i>False</i> then there will be no column headings. This property is affected by <code>SplitPropsGlobal</code> . If it is <i>True</i> , then headings in all splits are affected; if <i>False</i> then only headings in the current split are affected. If one split has headings and another does not, then the splits without headings show a "grey area" where the headings would be.
HeadingHeight	Specifies the height of the heading area, in lines. This number can be fractional. For example, 1.5 indicates that the headings should be 1.5 times as high as a single line of text. Defaults to 1, and applies globally to all splits in the grid.

Group: Horizontal and vertical grid lines

These properties specify how grid lines should appear. Both properties support settings for no lines, single-width lines, and 3D lines.

Properties

HorzLines	Indicates whether horizontal grid lines appear. This setting is global to the entire grid. The setting of HorzColor specifies the color of the lines, and can be tailored to each split.
VertLines	Indicates whether vertical grid lines appear. This setting is affected by SplitPropsGlobal. If it is <i>True</i> , then the line setting applies to all splits; if <i>False</i> then only the current split is affected.

Group: Miscellaneous grid appearance properties

Properties

MarqueeStyle	This property indicates how the marquee appears. The marquee defaults to a dotted cell border. Options include solid cell borders, highlighted cell, highlighted row, and highlighted row with a raised current cell. You can also turn the marquee off using this property. This property is affected by SplitPropsGlobal. If it is <i>True</i> , then the marquee will be changed for all splits; if <i>False</i> , then only the current split is affected.
MarqueeUnique	If <i>True</i> , then there will be only one marquee, no matter how many splits are in the grid. Changing splits will hide the marquee in one split and show it in the current split. If <i>False</i> , then the marquee will be present in all splits. The default is <i>True</i> .
BorderStyle	This property specifies whether the overall grid has a border or not.

Group: Row and column numbering

These properties allow you to change and inspect the size of the grid (in rows and columns) as well as change the current cell pointer, left column, and top row values. Events are included to determine when the cell pointer changes.

The current row number is global to the entire grid. All splits within the grid will have the same current row number. However, the column number can be different for each split.

Properties

BottomRow	Returns the number of the bottom row visible within the current split. Splits are normally synchronized, so this value will be the same for all splits. However, if splits are unsynchronized (by changing their SplitGroup), the bottom row value may be different for each split. This property is read-only and is available only at runtime.
ColumnIndex	Allows you to inspect or change the current column index. Columns are numbered based upon their original order, even if the user rearranges them (ColumnOrder can be used to inspect the actual position on the display). Setting this property is affected by SplitPropsGlobal. If it is <i>True</i> , then changing this property changes the current column for all splits.
Columns	Returns the number of columns in the grid. All splits must have the same number of columns, but columns can be hidden in some splits but not in others. You can set this property to define the number of columns. Setting it to zero removes all of the columns.
LeftColumn	Returns the column number of the leftmost column in the current split. Setting this property will scroll the specified column so it is the leftmost. If SplitPropsGlobal is <i>True</i> , then setting this property will change the leftmost column in all splits.
LinesPerRow	Specifies the number of lines which are displayed in each row of the grid. This is an integer number (no fractions allowed) and applies globally to the entire grid.
ReferenceRow	Sets or returns the row number used for the ColumnText, CellRectLeft, CellRectTop, CellRectWidth, and CellRectHeight properties. ReferenceRow is set automatically when a RowChange event occurs, or when the row number is temporarily different from RowIndex, such as in the UnboundFetch or FetchAttributes events.
RightColumn	Returns the column number of the rightmost column in the current split. Read-only and available only at runtime.
RowIndex	Sets or returns the current row number for the grid. The row number is global. That is, all splits will have the same row index.
Rows	Returns the total number of rows in the grid. In database mode, this number may not always reflect the actual size (if the number of rows is unknown). You can set this property, but only in callback mode where the grid must know how many rows to display.
TopRow	Sets or returns the current top row number for the current split. Normally, splits are synchronized, so setting the TopRow property will affect all splits even when SplitPropsGlobal is <i>False</i> . However, if splits are unsynchronized (by setting SplitGroup), then the TopRow may be different for each split.

Events

CellChange	This event is triggered whenever a new cell becomes current.
ColumnChange	This event is triggered whenever a new column becomes current, or when the user moves to a different split which has a different current column than the current one.

LeftColChange	Triggered whenever the left column changes to a new value. If there are splits, then this event is also triggered if a user changes splits and a new column becomes the left column (due to the difference in the horizontal position of the splits).
RowChange	This event is triggered whenever a new row becomes the current row.
TopRowChange	Triggered whenever the top row of the current split changes due to scrolling or because the value of TopRow has been changed in code. If there are independent splits (with different SplitGroup numbers), then this event is also triggered when changing to another split which is scrolled to a different top row position in the database.

Group: Calculated, bound, and unbound columns

When in database mode, TrueGrid columns can be *bound*, *unbound*, or *calculated*. Bound columns are automatically fetched from the database based upon their ColumnField property and are automatically updated. Unbound columns are filled in by the programmer using the UnboundFetch event. Calculated columns include an expression (in ColumnExpression) which TrueGrid uses to automatically calculate the value of the column. All column types receive the Validate and Update events whenever data is updated. For unbound and calculated columns, updating may not be possible. In such cases, the column should not be editable by the user (it is up to the programmer to configure this).

These properties apply globally to all columns in the grid. In other words, a column cannot be bound in one split while being calculated in another.

Properties

ColumnExpression	Specifies the Visual Basic expression to be used for calculating the value of the column when the ColumnType indicates that the column is calculated. Ignored for other column types, but still maintained. The expression must be a valid expression. Read/write at runtime. At design time, can be set in the Layout editor.
ColumnField	Specifies the field name for a bound column when the ColumnType indicates that the column is a bound column. Ignored for other column types, but still maintained. Read/write at runtime. At design time, can be set in the Layout editor.
ColumnType	Specifies whether the column is bound, unbound, or calculated.

Events

UnboundFetch	Triggered whenever an unbound column is scrolled into view and data is required.
Update	Triggered <i>after</i> a column has been updated by the user.
Validate	Triggered <i>just before</i> a column has been updated. The proposed value for the new cell is provided, and can be changed if the data to be written is different (in format for example) from the data the user sees in the edited cell.

Group: Miscellaneous column information

Each column has properties which define its behavior. These properties define miscellaneous behavior. Some are global, while others are specific to a particular split.

Properties

ColumnName	This property defines the string which will be used for the heading of a particular column. This property is global to all splits; that is, the column must have the same heading in all splits.
ColumnOrder	The column numbers used in Visual Basic code remain constant as columns are moved. This property defines the actual order of the column as it appears in the displayed grid. The value can be changed to move the column from one position to another. This property respects SplitPropsGlobal. If it is <i>True</i> , then changing ColumnOrder will move the column to the same position in all splits.
ColumnStyle	This property defines the text justification, heading justification, and editability of a column. Each split can have different values for the same column. This property respects SplitPropsGlobal. If <i>True</i> , changing ColumnStyle affects the column wherever it appears.
ColumnVisible	If <i>True</i> , the indicated column will be visible. If <i>False</i> , the column will not be shown. When the user resizes a column to zero width, this property is automatically set to <i>False</i> . Columns may be hidden in one split while they remain visible in another. If SplitPropsGlobal is <i>True</i> , setting this property changes the state of the column in all splits.
ColumnWidth	This property contains the width of the column, in characters. Changing it will change the column width for the current split, or for all splits if SplitPropsGlobal is <i>True</i> .
EditMask	This property defines the Visual Basic Format\$ string which will be used to format the text of the indicated column. It defaults to an empty string (no formatting). Changing this property will change the formatting of the column in all splits, regardless of the setting of SplitPropsGlobal.

Group: Scrollbars

TrueGrid allows complete control over scrollbars. These properties can specify whether horizontal or vertical scrollbars are never shown, always shown, or shown only if necessary.

These properties respect SplitPropsGlobal. Setting these properties while SplitPropsGlobal is *True* affects the scrollbars in all splits, while setting them while SplitPropsGlobal is *False* affects the scrollbars of the current split only.

Properties

HorzScrollbar	Indicates whether horizontal scrollbars are shown, hidden, or displayed only when necessary.
VertScrollbar	Indicates whether vertical scrollbars are shown, hidden, or displayed only when necessary.

Group: Layout configuration

The grid layout consists of a wide variety of split and column settings. Settings can be customized at design time using the Layout editor, or saved and restored in their entirety at runtime using the Layout property.

Properties

Layout

At design time, clicking on this property brings up the layout editor, which allows customization of column settings and other grid configuration options. At runtime, the Layout property contains a string which defines the complete layout of the grid, including splits, columns, column settings, etc. You can save the contents of the Layout property in a string and restore it later by reassigning it to the Layout property.

The format of the string is internal, and is checked for validity upon assignment. The string may contain null characters, so be careful when storing it in a file or INI file.

LayoutIndex

You can use this property at design time to set up multiple layouts in the grid. By changing the index number, the grid can be configured to a completely different layout. At run time, by switching numbers it is possible to show different layouts. You can also save changes the user makes to an existing layout.

Group: User keyboard and mouse navigation

Windows defines standard conventions for how the keyboard and mouse should operate. By default, TrueGrid conforms to these conventions. The tab key will move from cell to cell. At the first or last cell, tabbing moves out of the grid to the next or previous control on a form. Arrow keys maintain focus within the grid.

These properties are global to the entire grid and affect the way TrueGrid handles the arrows, tab key, and the mouse.

Properties

AllowArrows	If <i>True</i> (the default), then TrueGrid will respect the arrow keys for movement from cell to cell. If <i>False</i> , then the Arrow keys will always have their default meaning (moving from control to control).
AllowTabs	If <i>True</i> (the default), then TrueGrid will respect the tab key for movement from cell to cell. If <i>False</i> , then the tab key will always move between controls.
ExposeCellMode	Controls the behavior of the rightmost column when the user clicks on a cell in that column. When set to 0 (the default) the grid will scroll to the left to make the cell visible so all of its text can be shown. If set to 1 then the grid will not move, however if the user attempts to edit the cell then the grid will scroll left so you can see all of the cell text. If ExposeCellMode is set to 2 then the grid will always leave the last cell clipped both when clicked and when the user attempts to edit the cell.
MousePointer	Defines the appearance of the mouse cursor when it is positioned over the grid.
SplitSelectable	If <i>True</i> , then the current split will be selectable by the user with the mouse, and the tab and arrow keys can be used to move into the split (providing SplitTabMode allows such behavior). If <i>False</i> , then the user will not be able to select the split by clicking on it, and the split will be bypassed when moving from split to split with the keyboard. Changing this property while SplitPropsGlobal is <i>True</i> will affect all splits, otherwise only the current split will be affected.
SplitTabMode	When set to 0 (the default), the tab and arrow keys will not cross split boundaries. When set to 1, the tab and arrow keys will move to adjacent splits rather than be confined to the current split. This property is global to the grid and is not affected by SplitPropsGlobal.
TabCapture	If <i>False</i> , then the tab key will exit the grid and move to another control if an attempt is made to tab past the last column or before the first. If <i>True</i> , then the tab key will never exit the grid once it has focus. The default for this property is <i>False</i> .
TabIndex	Specifies where the grid lies in the tab order of the current form.
TabStop	If <i>True</i> , then the tab key will stop at the grid. If <i>False</i> , the tab key will bypass the grid and move to the next control in the tab sequence.
WrapCellPointer	If <i>True</i> , then the cell pointer will wrap from the last column to the first in the next row (or the first column to the last in the previous row). This will always operate with the arrow keys if AllowArrows is <i>True</i> . It will work with the tab key only if TabCapture is set to <i>False</i> , meaning that the tab key will not exit the grid. If SplitTabMode is set to 0, the cell pointer will wrap only within the current split. If SplitTabMode is set to 1, then the cell pointer will move from one split to the next before wrapping occurs.

Events

Click	Triggered whenever the user clicks on a cell within the grid.
DbClick	Triggered whenever the user double clicks on a cell within the grid.

Group: User modification of grid appearance

By default, the user can change the column order, size, and can create new splits and resize them. These properties control the users ability to do so.

Both properties respect `SplitPropsGlobal`. Setting these properties while `SplitPropsGlobal` is *True* affects the configurability of all splits, otherwise only the current split is affected.

Properties

Configurable	If <i>True</i> , then the user may resize columns, rearrange them, and hide them. If <i>False</i> , the column positions is fixed. In addition, when this property is <i>False</i> , it is possible to detect a click within the heading area (normally, the heading area is used to rearrange columns).
SplitLocked	If <i>True</i> , then the current split may be resized. If <i>False</i> , the current split cannot be changed. If all splits are locked, then the user will be unable to create any new splits interactively.

Group: Editing cell contents

TrueGrid automates the process of editing cell data. Cells are either in display mode or edit mode, depending upon whether the user has begun typing in a cell or whether the cell has been selected with the mouse. If the text is larger than the cell, TrueGrid will "drop-down" an editing box for editing the cell. In this mode, the arrow keys are redefined to move within the drop-down editing area.

Properties

ColumnChanged	This property is read-only and available only at runtime. If <i>True</i> , then the data in the specified cell has been updated by the user or through code modification. If <i>False</i> , then the cell contains the original data.
ColumnSize	This property restricts user input to a specified number of characters. If set to zero, then no limit is imposed. This property is global to all splits. That is, changing the value affects the data entry limit for the column wherever it appears. At design time, this property can be changed in the layout editor.
ColumnText	<p>This property contains the textual value of the specified column. The value of this property will contain the <i>unedited</i> contents of the cell (without the EditMask applied). Changing this value will cause the cell contents to be changed as if the user had modified the cell directly. Since a cell contains the same value regardless of which split it appears in, this property does not respect the SplitPropsGlobal setting.</p> <p>Normally, ColumnText refers to data within the current row. However, when used within the FetchAttributes, UnboundFetch, or FetchMargins events, this property will reflect the value of the row being fetched, regardless of whether it is the current row or not. When used in the context of these events, the ColumnText property cannot be changed (any change to the property is ignored).</p>
Editable	If <i>True</i> , then the grid is editable. If <i>False</i> , then edits are not allowed. This property is global to the entire grid.
EditActive	If this property returns <i>True</i> , then the current cell is currently being edited by the user. If <i>False</i> , then no editing is in progress. Setting this property to <i>True</i> will cause editing to begin on the current cell (if it is not already in edit mode). Setting it to <i>False</i> will exit edit mode. If the cell has been modified, this causes the Validate and Update events to be set. If you want to cancel editing, set the Modified property to <i>False</i> and then set EditActive to <i>False</i> .
EditDropDown	If <i>True</i> (the default) then the grid will automatically drop-down an editing box to allow editing of text which won't fit in the current cell. If <i>False</i> , then editing is always constrained to the current cell boundaries.
Modified	This property is useful only while the current cell is opened for editing. In such cases, Modified returns <i>True</i> the user has changed the data in the current cell and <i>False</i> if the data has been unchanged. The value of this property can be changed at runtime to change the modified status of the editing cell.
SelLength	Allows you to set or inspect the length of the currently selected text within the editing cell.
SelStart	Returns the start position of the selection in the editing cell. If SelLength is zero, returns the position of the caret.
SelText	Contains the selected text within the current cell being edited. Can be set to overwrite the selected text.
SplitEditable	If <i>True</i> , then the current split may be edited. If <i>False</i> , then editing is disallowed within the current split. If SplitPropsGlobal is <i>True</i> , then setting this property affects the editability of all splits, otherwise only the current split is affected. If the Editable property is <i>False</i> , then this property is ignored, but still maintained.
Text	Contains the text value of the cell being edited, otherwise returns an empty

string. Setting this property will change the contents of the current cell editing area and cause the grid to enter edit mode if it is not already editing the current cell. Changing this property does not cause the RequestEdit event to be triggered.

Events

Change	This event is triggered each time the user makes a change to the current cell being edited. The event is triggered for each character typed, just as for standard Visual Basic text boxes.
EnterEdit	This event is triggered once the current cell has been opened for editing. In other words, the EditActive property will always be <i>True</i> when this event executes. Although editing can be cancelled by setting EditActive to <i>False</i> in this event, the recommended method is to use the RequestEdit event.
ExitEdit	This event is triggered after the current cell has completed the editing process. If editing is cancelled (because the user presses escape, or because EditActive is set to <i>False</i> on an unmodified cell), the event is not triggered.
RequestEdit	Whenever the user attempts to edit the current cell, this event is triggered. The Cancel parameter can be set to <i>True</i> to cause the edit request to be ignored. Editing can begin either by clicking with the mouse, or by typing into a cell. The KeyAscii parameter to this event will be zero if the mouse was used to begin editing, or will contain the ASCII keycode used to begin the editing process.
Update	This event is sent <i>after</i> the current cell has successfully been edited.
Validate	This event is sent when the user completes editing of the current cell. The validate event may cancel the update, or may change the updated value before it is written.

Group: Controlling database behavior

TrueGrid generally handles all database operation smoothly and automatically. The following properties are used to control the interface between Visual Basic and the bound TrueGrid control.

Properties

Active	If set to <i>True</i> , then TrueGrid respects all operations performed on the database using the Data Control or its Recordset. If <i>False</i> , then TrueGrid ignores any database movement or updates performed in Visual Basic code or in conjunction with other bound controls. However, even when <i>False</i> , TrueGrid will still fetch data from the database during scrolling or refresh operations.
DataChanged	This standard Visual Basic property will be <i>True</i> whenever data in the grid has been changed but not yet updated to the database. Setting it to <i>False</i> will discard any pending changes in the grid.
DataSource	This property, available only at design-time, specifies the data control to which the grid is bound.
DataSourceHwnd	A runtime, this property may be set to the HWND of a Data Control to bind TrueGrid to that control dynamically. It is possible to bind TrueGrid to a data control on another form using this property. This property cannot be assigned to if DataSource has been set to a valid Data Control at design-time.
FetchMode	Normally, TrueGrid fetches cell data one cell at a time. In some circumstances (such as with SQL servers), this mode may <i>appear</i> slow to the user, since each access will require more time. This property allows you to change the method TrueGrid uses to fetch data to specify per-row or per-grid fetching rather than per-cell fetching which is the default.

Events

Append	When the user is positioned to the last row of the grid, pressing the down arrow key will trigger this event. It is up to the programmer to perform an AddNew and Update to the database, which will be reflected in the grid.
--------	--

Group: Working with multiple row selections

TrueGrid supports marking of multiple rows in the grid. This capability is enabled by setting the `SelectMode` property and examining the list of `Bookmarks` maintained by the grid. Bookmarks can also be added to the `Bookmark` list to cause records to be marked.

Properties

<code>BookmarkCount</code>	Contains the number of rows currently selected, or the size of the <code>BookmarkList</code> . Setting this property to zero clears all marked rows. It cannot be set to any other value.
<code>BookmarkList</code>	This is a property array which contains a list of selected rows, identified by their bookmarks. Setting an item to an empty string unmarks the row and removes the bookmark from the list. You can add a bookmark by assigning a new bookmark to the item one past the end of the <code>BookmarkList</code> .
<code>SelectMode</code>	This property indicates whether selection is enabled, and whether the selected rows are visible. Each may be controlled independently. Although bookmarks are global to the grid, this property respects the <code>SplitPropsGlobal</code> setting. Setting <code>SelectMode</code> while <code>SplitPropsGlobal</code> is <code>True</code> will change the selection mode for all splits in the grid; otherwise, only the current split is affected.
<code>SelectZoneWidth</code>	You can use this property to set the width of the area that allows the user to interactively select multiple rows. The number represents the number of pixels from the left that the checkmark will appear in. Note that in this zone the users will be unable to change rows.

Events

<code>MarkChange</code>	This event is triggered whenever the user marks or unmarks a row by clicking in the left margin of the grid.
<code>QueryMark</code>	This event operates only in callback mode. It is triggered whenever the grid needs to know whether a row is currently selected. It is up to the programmer to keep track of the list of selected rows, while TrueGrid handles the display and selection process.

Group: Controlling splits

TrueGrid supports powerful multiple-split capabilities, which can be used for Excel-like user-controlled splits, or can be used to implement locked columns. Although all splits display the same data, most properties are specific to a split and can be set differently for each split. SplitPropsGlobal affects this behavior.

Properties

InsertSplit	Assigning an integer to this property will insert a new split before the split indicated by the integer value.
RemoveSplit	Assigning an integer to this property will remove the split indicated by the integer value.
SplitGroup	This property contains the group number for a split. All splits with the same group number are "synchronized" so that they scroll simultaneously. Changing the group number for a split will "unlink" it from other splits so that it may scroll independently in the vertical direction. Splits are always independent for horizontal scrolling.
SplitIndex	This property contains the number of the current split, and can be changed to make a different split active. Splits are numbered from 1 to the number of splits in the grid. At design time, changing this property while SplitPropsGlobal is <i>False</i> allows you to set properties for a specific split within the design-time layout.
SplitPropsGlobal	This property affects the behavior of a large number of TrueGrid properties. If <i>True</i> (the default), then all properties apply globally to all splits. If <i>False</i> , then selected properties are applied only to the split identified by SplitIndex (the current split). Most properties, including column properties, can be customized on a per-split basis.
Splits	Contains the number of splits currently present in the grid. Setting this property to one higher than its current setting creates a new split.
SplitSize	This property contains the width of the split. Its meaning is dependent upon the value of SplitSizeMode. It contains either (a) a scale factor for the split, (b) the number of columns the split must contain, or (c) the width of the split in twips.
SplitSizeMode	This property determines how TrueGrid decides upon the size of the split. It can be set to (a) size the split according to a scale factor, (b) force the split to always display an integral number of columns, or (c) size the split to a specified number of twips.

Events

SplitChange	This event is triggered whenever the SplitIndex is changed either by user interaction, or by changing the SplitIndex property directly in code. The event is fired only once if SplitIndex is changed in code.
-------------	--

Group: Drag and drop

Most of these properties and events implement the standard Visual Basic drag-and-drop facilities. See the Visual Basic documentation for these properties. In addition, TrueGrid implements additional features to simplify the implementation of drag-and-drop interfaces.

Properties

DragIcon	Contains the icon used when the grid is dragged to another location. Defaults to a shadow of the grid's rectangle.
DragMode	Indicates whether dragging of the grid is manual or automatic. The setting of this property does not affect the DragCell event, which is sent in all cases.

Events

DragCell	This event is triggered whenever TrueGrid detects an attempt to drag a cell away from the grid. The cell does not need to be the current cell. The row, column, and split numbers of the dragged cell are provided as arguments. The programmer may perform any action in this event.
DragDrop	Indicates that another control has been dropped into the grid.
DragOver	Indicates that another control is being dragged over the grid.

Group: Determining cell dimensions and location

Normally, the programmer doesn't care exactly where cells are positioned relative to the coordinates of the form. However, in some cases (such as in the RequestEdit event) it is useful to position other controls relative to the current cell or a column. These properties allow you to determine the exact location of a cell. In order to be valid, the cell must be visible. If the cell is not visible, these properties return 0.

Properties

CellRectHeight	If the current cell is visible, this property returns the height of the current cell in the coordinate system of its parent form or container. This property cannot be set.
CellRectLeft	If the current cell is visible, this property returns the left boundary of the current cell in the coordinate system of its parent form or container. This property cannot be set.
CellRectTop	If the current cell is visible, this property returns the top boundary of the current cell in the coordinate system of its parent form or container. This property cannot be set.
CellRectWidth	If the current cell is visible, this property returns the width of the current cell in the coordinate system of its parent form or container. This property cannot be set.
ColumnRectLeft	If the specified column is visible, this property returns the left boundary of the column in the coordinate system of its parent form or container. This property cannot be set.
ColumnRectWidth	If the specified column is visible, this property returns the width of the column in the coordinate system of its parent form or container. This property cannot be set.
CurCellVisible	This property returns <i>True</i> if the current cell (indicated by RowIndex and ColumnIndex) is visible within the displayed area of the current split. It returns <i>False</i> if the cell is not visible. Setting this property to <i>True</i> will scroll the grid so that the cell is in view. Setting it to <i>False</i> is meaningless and is ignored.

Group: Finding cells based upon coordinates

TrueGrid supports several properties which can be used to determine, given a point on the grid, what column, row, or split is present at that point. These properties are especially useful for drag-and-drop applications, which need to know where in the grid data is to be dropped.

Properties

ColumnAtPoint	Returns the column which is beneath the point specified by PointX and PointY. If there is no column at that position, then this property returns zero.
PointX	Specifies the X coordinate (always in twips) of the point used by ColumnAtPoint, RowAtPoint, or SplitAtPoint. This property must be set by the programmer before asking for the Column, Row, or Split.
PointY	Specifies the Y coordinate (always in twips) of the point used by ColumnAtPoint, RowAtPoint, or SplitAtPoint. This property must be set by the programmer before asking for the Column, Row, or Split.
RowAtPoint	Returns the row number which is beneath the point specified by PointX and PointY. For the headings, zero is returned.
SplitAtPoint	Returns the number of the split which is beneath the point specified by PointX and PointY. If there is no split at the specified coordinate, zero is returned.

Group: Programmer control

These are miscellaneous properties and events which are mostly of use to the programmer when customizing the behavior of TrueGrid.

Properties

About	Available only at design time. Displays an About Box which indicates the current version of TrueGrid.
DataMode	Indicates whether the grid is being used in database mode, or callback mode (in which case data must be provided in the Fetch event, and the Rows and Columns properties must be set by the programmer).
Enabled	If <i>True</i> , then user interaction with the grid will be possible. If <i>False</i> , the user cannot interact with the grid, but it will still display data and respond to the database.
ExtendRightCol	If <i>True</i> when the the vertical scroill bar is not needed to display all the columns this property will extend the rightmost column to the grids edge. If <i>false</i> then the grid will respect the rightmost columns width setting.
Height	Contains the height of the grid in the scale mode of its container.
HelpContextID	Contains the context ID which will be used for the grid if context sensitive help is triggered by using the F1 key at runtime.
Hwnd	This read-only property will contain the Windows HWND handle of the grid. Although the grid supports complex visual behavior, it consists of a single Windows HWND Window. Users of this property generally must be familiar with the Windows API in order to use it.
HwndEdit	When editing is in progress, the grid displays a Windows edit control above the current cell. This read-only property will contain the HWND handle of that window. If editing is not in progress (<i>EditActive</i> is <i>False</i>), this property returns zero.
Index	The index of the grid, if it is part of a property array.
Left	The left coordinate of the grid in the scale mode of its container.
Name	The name used to identify the grid in Visual Basic code and at design time.
Parent	This property returns the control (a form, frame, or picture control) in which the grid is contained.
RefreshColumn	You can only write to this property, and then only at runtime. Setting this property to zero refetches all of the data in the current column of all splits. Setting it to a number other than zero refetches data only in the column indicated.
RefreshRow	You can only write to this property, and then only at runtime. Setting this property to zero refetches all of the data in the current row of all splits. Setting it to a number other than zero refetches data only in the row indicated.
Tag	This is a programmer defined text string which can be used for any purpose.
Top	The top coordinate of the grid in the scale mode of its container.
Version	This property returns the current internal version number of the TrueGrid control being used. The version number is an integer which is incremented for each distributed version, including patches and special releases.
Visible	If <i>True</i> , the grid will be visible to the user. If <i>False</i> , the grid will not be visible. Invisible grids still respond to the database unless their <i>Active</i> property is set to <i>False</i> .
Width	The width of the grid in the scale mode of its container.

Events

CheckRows	Enables the programmer to dynamically change the rows property during grid
-----------	--

	fetching.
Fetch	Called whenever the grid needs data to fill a cell in callback mode. Not used in database mode.
GotFocus	Triggered whenever the grid receives focus.
KeyDown	Triggered whenever a key is pressed in the grid, or in the editing cell. This event is triggered <i>before</i> the grid responds to the key, but it cannot be cancelled. Use the Form KeyPreview property to capture keys before the grid receives them.
KeyPress	Triggered when an ASCII key is pressed. The keypress can be cancelled if you don't want the grid to respond to the key.
KeyUp	Triggered when a keyboard key returns to the unpressed state.
LostFocus	Triggered whenever the grid loses focus.
MouseDown	This event is triggered when either mouse button goes down and is triggered <i>before</i> the grid receives the mouse event.
MouseMove	Triggered when the user moves the mouse anywhere inside the grid.
MouseUp	Triggered when either mouse button returns to its unpressed state.
Paint	This event is triggered whenever the grid repaints itself. This occurs frequently in the Windows environment and is generally useful only for special circumstances. In this event, users familiar with the Windows API may use the Hwnd of the grid (returned by the Hwnd property) to paint special effects, including lines, bitmaps, icons, etc. in appropriate cells of the grid.

Group: Value list

These are properties and events that relate to using TrueGrid's value list features at run time.

Properties

ColumnButton	Determines whether or not a dropdown button will appear in a given column.
ColumnComboMaxitems	Determines how many items will be displayed in the grid's automatic drop down combo box, before a scroll bar will appear.
VlistColumn	Sets the current value list column for which all value list properties are being read from or written to. Similar to SplitIndex or ColumnIndex and their respective properties.
VlistCount	Returns the number of entries in the value list. Can be set to zero to remove the value list for the current value list column.
VlistData	An indexed property that returns the specific items in the value list for the data column. Corresponds to the Data column In the Field Values Dialog.
VlistDefault	Specifies a default value to be used for the current value list column. Any value that is not in the value list use the default display value instead.
VlistPicture	If you want to display a picture in place of the value currently displayed then set this property for the corresponding value in the VlistData property and have the ColumnStyle property set with GVLS_TRANSLATE.
VlistStyle	Determines what options will be used for the column's value list. This property allows you to have the list use translated values, cycle through available option, auto validate, and how the data is to be displayed.
VlistText	If you want a string in place of the value currently displayed then set this property for the corresponding value in the VlistData property and have the ColumnStyle property set with the GVLS_TRANSLATE.

Events

ClickButton	This event is triggered whenever a column with the ClickButton property set to <i>True</i> has the button clicked by the user. The event passes the current column. Useful for popping up other controls.
ValidateError	This event is triggered only when the VlistStyle is set to GVLS_VALIDATE. If the user enters a value that is not in the value list then the ValidateError is fired. Use this event to determine what to do with new or incorrect values.

Group: Column Summary

These are properties and events that relate to using TrueGrid's column summary features.

Properties

ColumnSum	Enables the programmer to set the seed of the current column's sum. You can also read this property at any point to get the current column's sum.
ColumnSumEnable	Boolean value that turns on a column's summary feature. If set to <i>True</i> then the grid will keep a sum for that column. If <i>False</i> then the grid will not sum the column.

Events

ColumnSumChange	While the grid does keep the sum internally the programmer may want to constantly display that value. When there is a change in the column sum then this event is triggered.
-----------------	--

