Get Rid Of File Maintenance Gridlock With Grid Control

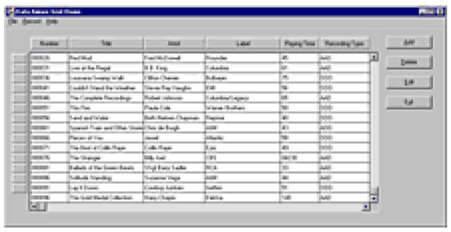
By Kurt Williams and Amy Petré Hill

Editors Note: *The following is the first of a two part tutorial, based on an online tutorial currently offered in the free Visual PRO/5® 2.02 update, that demonstrates exactly how a developer can unleash the power of the grid control to create GUI, data-aware file maintenance interfaces. The data used in this tutorial comes directly from data provided in the* **`datagrid.dat`** *file, also provided in the Visual PRO/5 2.02 update. The complete Datagrid coding sample can be downloaded from BASIS' Visual PRO/5 2.0 Coding Samples web page at www.basis.com/visualpro5/samples.html.*

Part two of this tutorial is also available online.

Ask Business Basic developers to list their ten *least* favorite programming tasks and chances are that they'll put file maintenance near the top of the list. Although file maintenance itself is not hard, the file maintenance utilities that developers must purchase or create to make file maintenance possible for end users have long presented problems.

Hours of precious programming time are regularly spent creating and then updating character-based file maintenance interfaces that work but do not possess the Windows "look and feel" that customers want. What developers need is some kind of interface tool that can tie into a file and display the information in an easy-to-read GUI format.


Complete data-aware grid.

With the release of Visual PRO/5 2.0, the Business Basic community finally has this kind of an interface tool available with the new grid control. A grid, laid out in columns and rows like a spreadsheet, can display data in a format that is both easy to move through and easy to read, while offering the popular "Windows look."

For small data sets, such as "Sales This Month," a developer can use the standard grid control to quickly set up the entire grid. Each row can represent a day, while the columns contain data for the daily sales, sales tax, cash in, credit cards received, payouts, as well as overages and shortages. The visible area of the grid can be large enough to show ten days at a time, and the scroll bars allow the end user to see all the columns and all the days in the month.

When larger volumes of data, such as an entire file, need to be maintained, the grid control can generate a data-aware grid capable of handling all the back-end manipulation and formatting of the data. A data-aware grid is a grid control bound to a channel that is opened to an MKEYED file, PRO/5 SELECT, or an SQL SELECT. The grid itself manages the presentation of the data while also handling any edits, deletions, or additions the user may make to the file.

Creating the Resource

In Visual PRO/5 2.0, data-aware grid development begins in the ResBuilder™ graphical resource development tool. Start ResBuilder™ and select `New Resource File` from the `File` menu to create an empty resource file. Then select `Add Form` from the `Edit` menu to create a blank form numbered 101. In its property sheet, leave all properties at their default values except as follows:

```
Title = Data Aware Grid Demo
Name = frmDataAwareGrid
x position = 100
y position = 100
Width = 875
Height = 400
```

The result is a form that is ready to accept the grid.

Create the grid control by clicking on the `Grid Control` tool button and then clicking anywhere on the face of the new form. In the grid's property sheet, set the following properties to the specified values:

```
Name = grdTestGrid
x position = 12
y position = 22
Width = 731
Height = 353
Num Rows=1
Num Columns=12
Row Head = checked
Row Head Width = 15
Col Head = checked
Horiz Scroll = checked
Vert Scroll = checked
```

The **`Num Rows`** property is set to **`1`** at design time, but when the grid is bound to the channel, the number of rows will be governed by the number of records in the file. The **`Num Columns`** property is set to **`12`** because this grid will be used to present a particular file that has twelve columns, and the widths of these columns need to be defined at design time. If no specific column width was needed, the default could simply be set to **`1`** in the **`Num Columns`** field. When the grid was bound to the channel at runtime, the number of columns would be set by template and the columns set to a default width. Using the SENDMSG() Function 36, the program could set the column width at runtime. Additionally, the user could set the column width using the mouse with the grid's visual interface.

Next, in the grid's property sheet there is a value called `Column Prop` which provides a button with an ellipsis (…) on it. Clicking this button opens the Column Property dialog. Using this dialog, set the column width for each of the twelve columns as follows (leave the Column Title field blank for all columns):

```
Column Number 1 set Column Width = 75
Column Number 2 set Column Width = 150
Column Number 3 set Column Width = 150
Column Number 4 set Column Width = 150
Column Number 5 set Column Width = 75
Column Number 6 set Column Width = 100
Column Number 7 set Column Width = 75
Column Number 8 set Column Width = 75
Column Number 9 set Column Width = 100
Column Number 10 set Column Width = 75
Column Number 11 set Column Width = 75
Column Number 12 set Column Width = 75
```

Now add four buttons as follows by clicking on the Push Button tool button and clicking on the face of the form. In the property sheet for each button, set the following values:

```
First Button
    Name = btnInsert
    Text = &Add
    x position = 758
    y position = 20

Second Button
    Name = btnDelete
    Text = &Delete
    x position = 758
    y position = 60

Third Button
    Name = btnEdit
    Text = &Edit
    x position = 758
    y position = 100

Fourth Button
    Name = btnExit
    Text = E&xit
    x position = 758
    y position = 140
```

Save the resource by selecting the Save As option from the File menu. Give it a name in the file save dialog. The resource is now ready to become part of our file maintenance program.

Creating the Program

Start GUIBuilder™ and select New in the File menu to create a new GUIBuilder file. In the Name New GUIBuilder File dialog, give the file a name. To simplify, keep all the files together by giving the GUIBuilder file the same name as the resource you just created.

When the Create New Resource message box pops up, click No--the appropriate resource has already been created in ResBuilder. Then select the resource you created and click Open. This will be followed by the Program Options dialog that sets the options for the program about to be generated. Click OK to accept the default values. Now the resources are ready and the grid development can begin.

First select End of Job Code from the Object drop-down list box. The End of Job header remarks will appear in the GUIBuilder edit area. Type **release** on a new line after the remarks. The end of job code is executed when the program terminates. In this example, Visual PRO/5 2.01 will be released.

To make the Exit button functional, select Form 101 frmDataAwareGrid from the Object drop-down list box. In the Control drop-down list box, select Push Button 106 btnExit. In the Event drop-down list box select Button Pushed. Forms and controls may have different ids, but if the above naming suggestions were followed, the names should be the same. In the GUIBuilder edit area, insert the following line of code directly below the remark that says **`rem 'Push button operated'`**:

```
gb__eoj=1
```

This code will be executed when the user clicks on the Exit button while the program is running. It sets the GUIBuilder exit flag to true and causes the event loop to terminate and execute the End of Job code. *Please note that GUIBuilder special variables and function names are always indicated with* **`gb`** *followed by two underscores.*

Now select Run Program from the Run menu. Click Save in the Save Program As dialog to save the generated program with the same name as the project. The program will run, displaying the resource created. Although the grid does not do much yet, it does show how the resource will look at runtime. Click the Exit button and the program will shut down, releasing the secondary copy of Visual PRO/5 that GUIBuilder started to run the grid.

Making the Grid Data-Aware

For this example, use the file called **`datagrid.dat`** provided with the Visual PRO/5 2.02 release. To get the project ready to present a grid bound to a file, some initialization code and three subroutines must be written.

First, write the initialization code. Begin by selecting Initialization Code from the Object drop-down list box. The Initialization header remarks will appear in the GUIBuilder edit area. Below the remark headers, add the following code:

```
rem get a template describing the form using
: GUIBuilder's get template function
dim datagrid_temp$:fngb__template$(gb__win_id$)

rem setup constants
gosub define_constants

rem open the data file and setup the template
gosub open_data_file

rem make the grid data aware
gosub bind_grid_to_chan
```

The code above uses a GUIBuilder-provided function to retrieve a template that describes the resource. It will be used to get the control id of the grid. It then executes **`GOSUB`**s to three different subroutines that will be written in the following three steps.

To create the **`define_constants`** routine, select New Subroutine/Function from the Object drop-down list box. In the Name Subroutine/Function dialog, type **`Define Constants`** and click OK. A set of remark headers will appear in the GUIBuilder edit area.

Enter the following code after the initial comment block:

```
define_constants:

rem message box constants
    msgboxYes=6
    msgboxYesNo=4
    msgboxExclamation=48
    msgboxInfo=64
    msgboxSecond=256

rem grid send message functions
    gridSetHeadingTitles=23
    gridEndEdit=26
    gridStartEdit=31
    gridGetEdit=34
    gridSetEdit=35
    gridGetNumberofCols=40
    gridGetNumberofRows=41
    gridGetSelectedCol=44
    gridGetSelectedRow=45
    gridGotoCol=47
    gridGotoRow=48
    gridShowCurrentHeading=77
    gridSetDataAware=80
    gridDataAwareFunctions=81

rem misc grid values
    gridHeadingDepressedMode=1
    gridHeadingNotDepressedMode=0

rem data aware functions
    gridSetReadOnly$=$01$
    gridDeleteRow$=$02$
    gridAddRow$=$03$
    gridRetrieveRow$=$04$
    gridCancelUpdate$=$05$

return
```

This subroutine creates a series of variables that will be used in the grid's SENDMSG() functions to make the code more readable.

To create the **`open_data_file`** routine select New Subroutine/Function from the Object drop-down list box and in the Name Subroutine/Function dialog, type **`Open Data File`**. Below the remark headers for the new routine, enter the following code:

```
open_data_file:

data_chan = unt
open(data_chan)"datagrid.dat"

rem an alternate channel used for file operations
rem this channel will not be bound to the grid
alt_chan = unt
open(alt_chan)"datagrid.dat"

rem set up the template
datarec_desc$="CDNUMBER:C(6*=10):SHOW=1 ALIGN=0" +
: " LABEL=Number:," +
: "TITLE:C(50*=10):SHOW=1 ALIGN=0 LENGTH=50" +
: " LABEL=Title:," +
: "ARTIST:C(50*=10):SHOW=1 ALIGN=0 LENGTH=50" +
: " LABEL=Artist:," +
: "LABEL:C(50*=10):SHOW=1 ALIGN=0 LENGTH=50" +
: " LABEL=Label:," +
: "PLAYINGTIME:C(6*=10):SHOW=1 ALIGN=0 MASK=000.00" +
: " LABEL=Playing_Time:," +
: "RECORDINGTYPE:C(3*=10):SHOW=1 ALIGN=0 MASK=AAA" +
: " LABEL=Recording_Type:," +
: "MUSICTYPE:C(15*=10):SHOW=1 ALIGN=0 LENGTH=15" +
: " LABEL=Music_Type:," +
: "BINLOCATION:C(10*=10):SHOW=1 ALIGN=0 LENGTH=10" +
: " LABEL=Bin_Location:," +
: "NUMBEROFTRACKS:N(10):SHOW=1 ALIGN=1 MASK=0000" +
: " LABEL=Number_of_Tracks:," +
: "ONHAND:N(10):SHOW=1 ALIGN=1 MASK=000000" +
: " LABEL=On_Hand:," +
: "COST:N(10):SHOW=1 ALIGN=1 LABEL=Cost:," +
: "RETAIL:N(10):SHOW=1 ALIGN=1 LABEL=Retail:"
dim datarec$:datarec_desc$

return
```

This code opens the file on two channels: one for use by the grid and one for use by the program in creating new keys. It also creates a template description in **`datarec_desc$`** and the template itself in **`datarec$`**. Note the user attributes for each field. They control how the grid will look and behave.

The **SHOW** attribute controls whether the field will be shown. **SHOW=1** indicates the field should be shown. **SHOW=0** will hide the column.

ALIGN controls the alignment of the data within the grid cell. **ALIGN=0** means left justify the data. **ALIGN=1** means right justify the data. **ALIGN=3** causes the data to be centered.

LABEL provides the text for the column headers. If a space is needed in the label text, use the underscore character ("_"). It will be replaced with a space when displayed in the column header.

MASK provides an input mask that will be used when the cell is placed in edit mode. Valid mask characters are the same as those used by INPUTE. When a cell is placed in edit mode, the grid uses a special INPUTE control to do the editing.

To create the **bind_grid_to_chan** routine, select New Subroutine/Function from the Object drop-down list box. In the Name Subroutine/Function dialog, type **Bind Grid to Chan** and click OK. Below the header remarks in the GUIBuilder edit area, enter the following code:

```
bind_grid_to_chan:

rem get the grid id from the template describing the form
grid_id=num(fattr(datagrid_temp$,"grdTestGrid","ID"))

rem send message 80 (gridSetDataAware) to grid to
rem bind it to the channel
tf$=sendmsg(gb__sysgui,grid_id,gridSetDataAware,
: data_chan,datarec_desc$)
tf$=sendmsg(gb__sysgui,grid_id,gridShowCurrentHeading,
: gridHeadingDepressedMode,$$)

return
```

The line after the first remark extracts the grid id number from the template that describes the form. This template was created in the initialization code.

The line after the second remark binds the grid to the channel. This is a grid SENDMSG() Function 80. The parameter list within the SENDMSG() function contains several variables. The channel on which GUIBuilder opened the SYSGUI is called **gb__sysgui**. The control id of the grid is **grid_id**. The constant defined in **define_constants** is **gridSetDataAware** and is equal to 80. The channel on which the file opened is **data_chan** and **datarec_desc$** is the template description we created in **open_data_file**.

The last line uses grid SENDMSG() Function 77 to set the headings, both row and column, to be displayed as depressed for the current row and column as the user navigates through the grid.

The program is now ready to run as a data-aware grid.

To try out the new grid, select Run Program from the Program menu and click the Save button in the Save Program dialog. The program will present the records from the file in the grid. Use the vertical scroll bar to move up and down the file. Click with the mouse in various cells. This will move the grid focus to the clicked cell. Notice how the column and row headers for the highlighted cell are depressed.

In the next issue of *The BASIS Advantage*, the second part of this tutorial will complete the grid control demonstration by showing how a developer can add more functionality to the grid with the introduction of the Edit, Add, and Delete functions.